Contents

```
% Process CHN runs from Elementar Cube analyzer
```

```
clearvars
close all
tic

% Runs Info

cruise = 'P06_2017';   %enter here cruise or campaign code
```

Settings

```
calQA = 'yes'; % assess quality of cal runs and remove suspect ones

exportFigs = 'no';

LineWidths = 1.6;

newColors = get(groot,'DefaultAxesColorOrder');

scaledMADs = 3.5;  % number os scaled Median Absolute deviations for...
                   % calibration run to be classified as an outlier

if ispc
    homeStr = 'C:/Users/jchavesc';
else
    homeStr = '~';
end
```

Identify all files with runs

```
path2Data = [homeStr '/Documents/CVO/Biogeochem/POC/P06_Runs/'];

cd(path2Data)

if ispc
    delete('._2*') % removes hidden files created by Excel
else
    system('rm ._2*') % removes hidden files created by Excel
end

runFiles = dir('*.xlsx');
```

```
rm: cannot remove '._2*': No such file or directory
```

```
ans =

     1
```

Process each run file

```
for k = 1:length(runFiles)
```

Data info

```
        rundID = num2str(k); % run ID (temp) *****

        runFileStr = strsplit(lower(runFiles(k).name),'_fac'); % to tag plots
```

Open Data File that contains analysis report from CHN instrument

your data is located

```
        sheetStr = 'samples'; % sheet tab name

        % open data and put into numeric, text, and 'raw' outputs

        [num,txt,raw] = xlsread(runFiles(k).name,sheetStr);

        % use data from 'raw' import from XLS file
        % and remove header row

        data = raw(2:end,:);

        % extract 1st line of headers
        headers = txt(1,1:end);
        headers = string(headers); % convert to a vector of strings

        % Column for run 'Name'

        namesCol = headers == "Name";
```

Find index for each type of run

```
        runLength = length(data);

        % Preallocate vectors for run class indeces

        sulfDex = zeros(size(runLength));   % index of sulfanilamide standards
        airDex  = zeros(size(runLength));   % index of air blanks
        tinCalDex = zeros(size(runLength));   % index tin boats blanks
                                % (i.e.  for running weighed standards)
        tinSampDex = zeros(size(runLength));   % index tin circle blanks (i.e. for running filter with samples or filter blanks)
        buffRivDex = zeros(size(runLength));   % index Buffalo River Sediment reference runs
        sampleDex =  zeros(size(runLength));   % index of sample runs
        sampleSeq =  zeros(size(runLength));   % Sample ID

        % Preallocate for tin blanks means from all runs to fill runs
        % without blank runs

        if k == 1
            allCarbCalBlanks = zeros(k,1);
            allCarbSampBlanks = zeros(k,1);
            allNitroCalBlanks = zeros(k,1);
            allNitroSampBlanks = zeros(k,1);
        end

        % Column with names/ID entered for each run

        runNames = data(:,namesCol);

        for i = 1:runLength

            if isnumeric(runNames{i}) % if runNames{i} convert directly to number is a sample ID
```

```
                sampleDex(i) = 1;
                sampleSeq(i) = runNames{i};

            else

                runStr = runNames{i}; % string entered for individual run

                % Cleanout runStr for spaces and make it all lowercase

                runStr = lower(runStr(~isspace(runStr)));

                % clean NaNs

                if isnan(runStr)
                    runStr = 'empty';
                end

                % sulfanilamide
                sulfDex(i) = any(regexpi(runStr,'sulfa'));  % sulfanilamide standards i.e., any runStr containing 'sulfa...'
                sulfDex = logical(sulfDex);

                % air blanks
                airDex(i) = any(regexpi(runStr,'air'));  % air blanks i.e., any runStr containing 'air...'
                airDex = logical(airDex);

                % acidified filter blanks
               %  acidFiltDex(i) = any(regexpi(runStr,'acid'));  % sulfanilamide standards i.e., any runStr containing 'sulfa...'
               %  acidFiltDex = logical(acidFiltDex);

                % Tin blanks (boats and sheets)

                    if any(regexpi(runStr,'tin'))

                        if any(regexpi(runStr,'35x35')) || any(regexpi(runStr,'30x30'))
                            tinSampDex(i) = 1;
                        else
                            tinCalDex(i) = 1;
                        end
                    end

                     tinSampDex = logical(tinSampDex);
                     tinCalDex = logical(tinCalDex);

                    % Sample runs numbers stored as string

                     if ~isnan(str2double(runStr))  % if run string converts to number

                         sampleDex(i) = 1;
                         sampleSeq(i) = str2double(runStr);

                     end

                    % Buffallo River NIST reference

                    buffRivDex(i) = any(regexpi(runStr,'buff'));  % Buffallo River reference i.e., any runStr containing 'buff...'
                    buffRivDex = logical(buffRivDex);

            end
                    sampleDex = logical(sampleDex);
        end
```

Instrument signals for each element

Carbon: Carbon signal data.Area2

```
          % All carbon areas

            area2Col = headers == "Area2";

            carbonAreas = data(:,area2Col);          %  Carbon signals (i.e. Area2)

            carbonAreas = cell2num(carbonAreas);

            % calibration (tin boats) blanks
            calCarbBlankReps = carbonAreas(tinCalDex);  %  C blanks for calibration runs

            calCarbBlank = mean(calCarbBlankReps);      %  Mean C blanks for calibration runs ONLY

            allCarbCalBlanks(k) =  calCarbBlank;

            if isnan(calCarbBlank)
                calCarbBlank = nanmean(allCarbCalBlanks(1:k));
            end

            % Sample tin circle blanks

            sampleCarbBlankReps = carbonAreas(tinSampDex);  %  C tin blanks for sample runs

            sampleCarbBlank = nanmean(sampleCarbBlankReps);    %  Mean C tin blanks for sample runs

            allCarbSampBlanks(k) =  sampleCarbBlank;

            if isnan(sampleCarbBlank)

                sampleCarbBlank = nanmean(allCarbSampBlanks(1:k));

            end
```

Run CARBON calibration curves

```
            % Carbon -- Amount of C in mg for sulfanilamide runs

            weightCol = headers == "WeightVol";

            weights = data(:,weightCol);

            weights = cell2num(weights);

            sulfCarbWeight   = (41.8/100).* weights(sulfDex); % mg C per sulfanilamide run

            sulfCarbAreas  = carbonAreas(sulfDex) - calCarbBlank;  % Sulfanilamide areas, blank corrected

            if isnan(calCarbBlank) || isnan(sampleCarbBlank)
                stop
            end

         x = sulfCarbAreas;
         y = sulfCarbWeight;
```

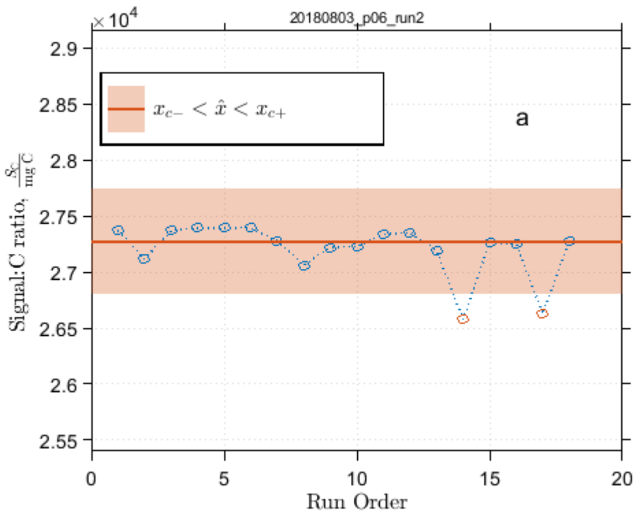
Calibration run QA

```
        if strcmpi(calQA,'yes')


            areaCarbRatio = x./y;  % area to carbon ratio in sulfanilamide stds
            medianCarbRatio = nanmedian(areaCarbRatio);
            stdCarbRatio = std(areaCarbRatio);
            absCarbResid = abs(areaCarbRatio - medianCarbRatio);
            maxCarbResid = max(absCarbResid);
            MADCarbRatio = mad(areaCarbRatio,1);
            modZScores = 0.6745 * ((areaCarbRatio - medianCarbRatio) ./ MADCarbRatio);

            % cal QA plot
            % Tweaks so that figures renders the same in Mac & Linux

            screens = handle(0); %
            mainScreen = screens.MonitorPositions(1,3:4);

            if ismac
                h0i = figure('Position',[mainScreen(1) * 1.06 378 364 636]);
            elseif isunix
                h0i = figure('Position',[mainScreen(1) * 1.06 209 500 900]);
                h0i.Renderer = 'OpenGL';
            end

            subplot(2,1,1)
```

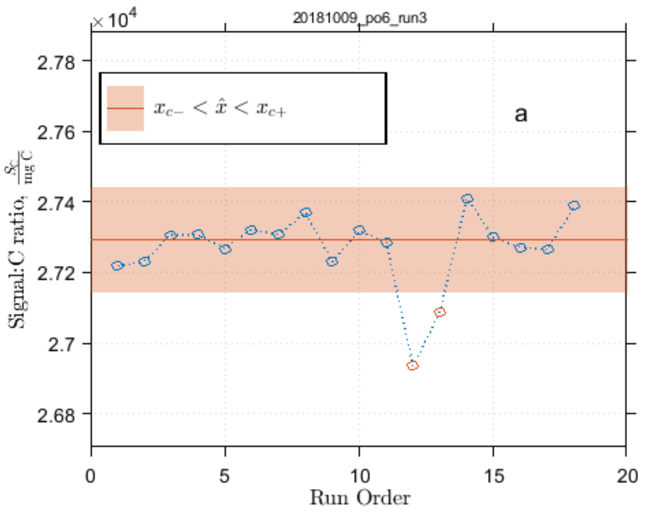
```
L1 = plot(areaCarbRatio,'o:');
hold on

% axes properties

h1 = handle(gca);
h1.XLim = [0 ceil(numel(areaCarbRatio) * 1.06)];
h1.LineWidth = LineWidths;
 grid on

h1.TickDir = 'out';
h1.TickLength = h1.TickLength .* 1.5;

L2 = hline(medianCarbRatio);
L2.LineStyle = '-';
L2.Color = newColors(2,:);

% An outlier is a value that is more than three SCALED median absolute ...
% deviations (MAD) away from the median:
%
% For a random variable vector A made up of N scalar observations,
% the median absolute deviation (MAD) is defined as

%     MAD' = median(abs(A-median(A)));
%
%  The scaled MAD is defined as c * MAD
%  where c = 1.4826 and is given by:
% -1/(sqrt(2)*erfcinv(3/2)).
%
%  [***NOTE***] In Matlab mad(x) gives the MEAN absolute deviation from
%  the media & while in R the function mad(x) gives the MEDIAN absolute
%  deviation from the median. To get the scaled MAD in Matlab use the
%  syntax mad(x,1)

% Graphical implementation of the above with the threshold values

X = [h1.XLim fliplr(h1.XLim)];

c = -1/(sqrt(2)*erfcinv(3/2)); %

M_i = scaledMADs * c * MADCarbRatio;

    y1 = medianCarbRatio - M_i;
    y2 = fliplr(medianCarbRatio + M_i);
Y = [y1 y1 y2 y2];
L3 = patch(X,Y,L2.Color);
L3.FaceAlpha = 0.3;
L3.EdgeColor = 'none';

% Y Lims

f = 4;
h1.YLim = [(medianCarbRatio - (M_i * f))  (medianCarbRatio + (M_i * f))];

%    L1 = plot(areaCarbRatio,'o:');

%hline([meanCarbRatio meanCarbRatio-stdCarbRatio  meanCarbRatio+stdCarbRatio],{'b', 'r', 'r'},{'mean+/-s.d.','',''})
xLab1 = xlabel('Run Order');
xLab1.Interpreter = 'latex';
yLab1 = ylabel('Signal:C ratio, $\frac{S_{\rm C}}{{\rm mg\, C}}$');
yLab1.Interpreter = 'latex';
%title(['C calibration Q/A ' rundID  ' run'])

% ID oulier Sulfanilamide runs (Mean )areaCarbRatio

goodCalRuns = abs(modZScores) < scaledMADs; % ~isoutlier(areaCarbRatio,'median',3);  % index of cal runs to keep
badCalRuns = abs(modZScores) > scaledMADs; % isoutlier(areaCarbRatio,'median',3);    % index of cal runs to keep

xx = 1:numel(areaCarbRatio);

L11 = plot(xx(badCalRuns),areaCarbRatio(badCalRuns),'o');
if ~isempty(L11)
    L11.Color = newColors(2,:);
end
```

```
            x = x(goodCalRuns);
            y = y(goodCalRuns);

            % Legends

            LegL3 = patch(h1.XLim(2) * [0.03 0.03 0.1 0.1],h1.YLim(1) + (range(h1.YLim) * [0.76 0.87 0.87 0.76]),L2.Color);
            LegL3.FaceAlpha = 0.3;
            LegL3.EdgeColor = 'none';
            LegL4 = plot([h1.XLim(2) h1.XLim(2)] .* [0.03 0.1],h1.YLim(1) + (repmat(range(h1.YLim),1,2) .* [0.815 0.815]));
            LegL4.LineWidth = LineWidths;
            LegL4.Color = L2.Color;
            ttL1 = text(h1.XLim(2) * 0.1167,h1.YLim(1) + (range(h1.YLim) * 0.815),'$x_{c-} < \hat{x} < x_{c+}$');
            ttL1.Interpreter = 'Latex';
            ttL1.FontSize = 18;

            tt01 = text(h1.XLim(2) * 0.8,h1.YLim(1) + range(h1.YLim * 0.8),'a');
            tt01.Units = 'Inches';
            tt01.Position = tt01.Position;
            tt01.FontSize = 20;

            % Box
            bx = patch(h1.XLim(2) * [0.018 0.018 0.55 0.55],h1.YLim(1) + (range(h1.YLim) * [0.73 0.9 0.9 0.73]),'w');
            bx.LineWidth = LineWidths;

            %Sort Graphic objects

        %     h1.SortMethod = 'childorder';
        %
        %     hP = handle(h1.Children(8));
        %
        %     uistack(hP,'bottom');

        ti01 = title(runFileStr{1});
        ti01.FontWeight = 'normal';
        ti01.Interpreter = 'none';
        ti01.FontSize = 12;

        end
```

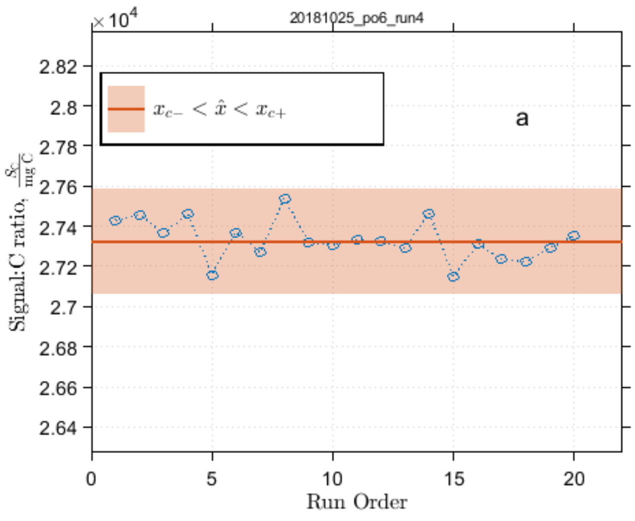
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



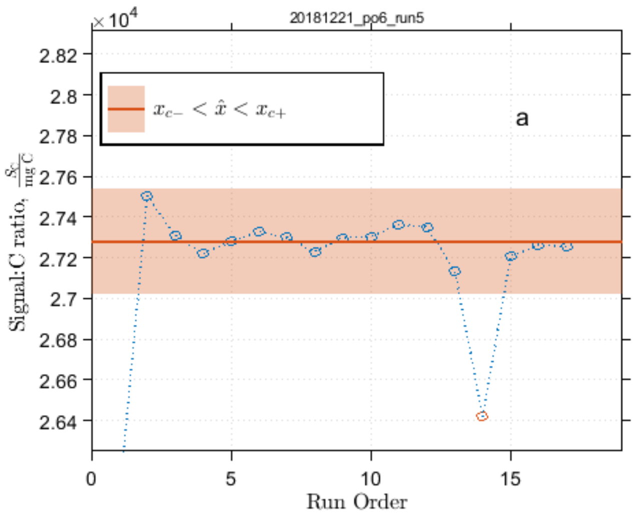
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



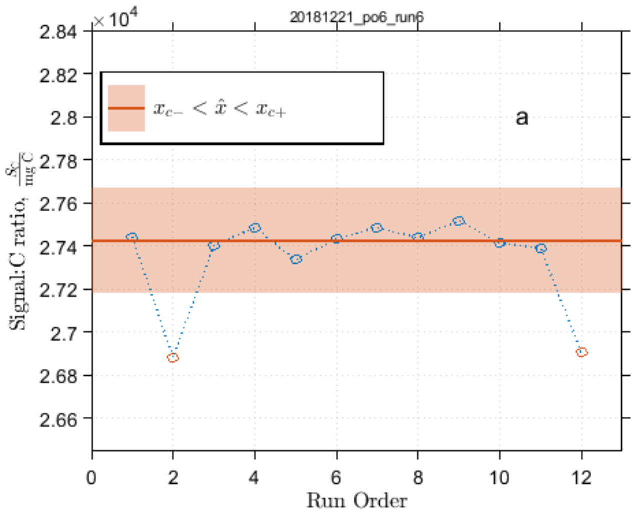
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



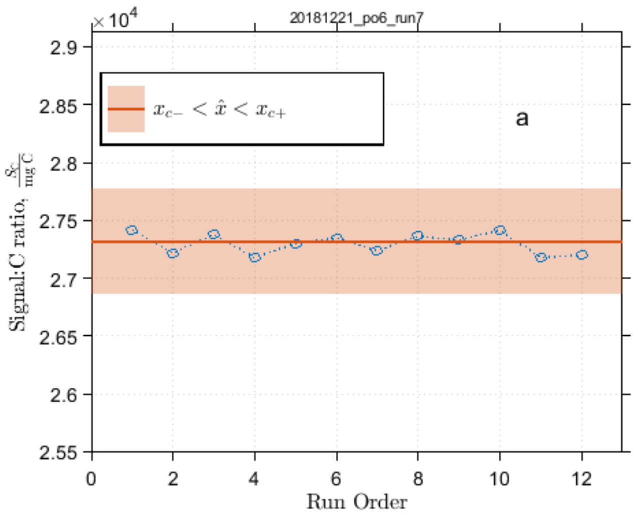
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



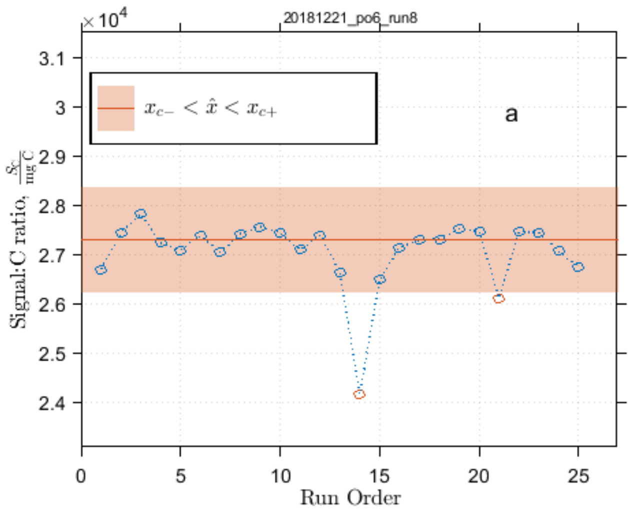
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



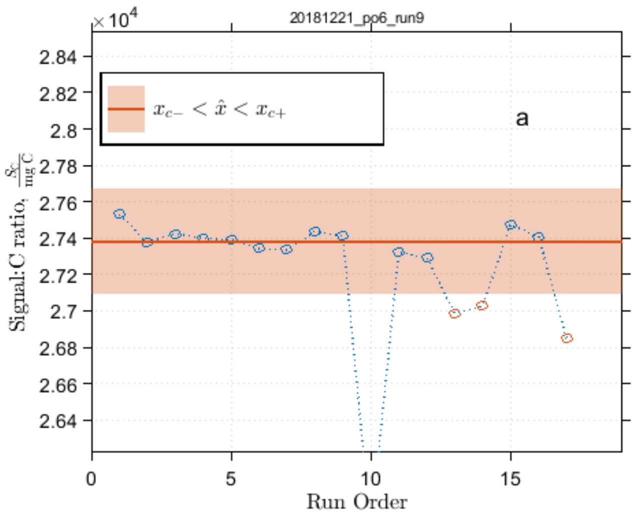
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



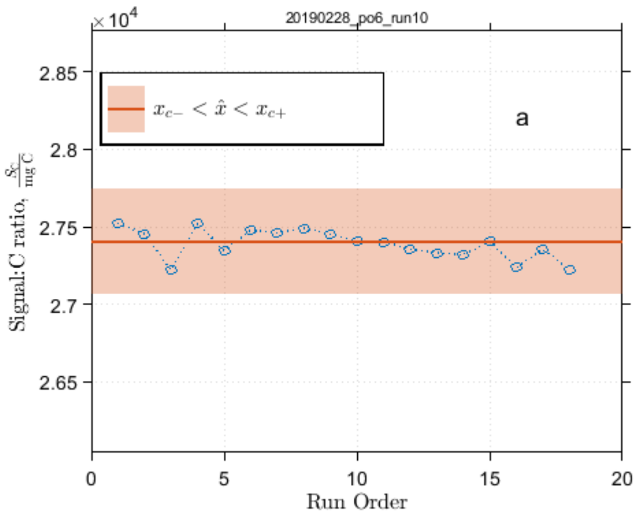
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



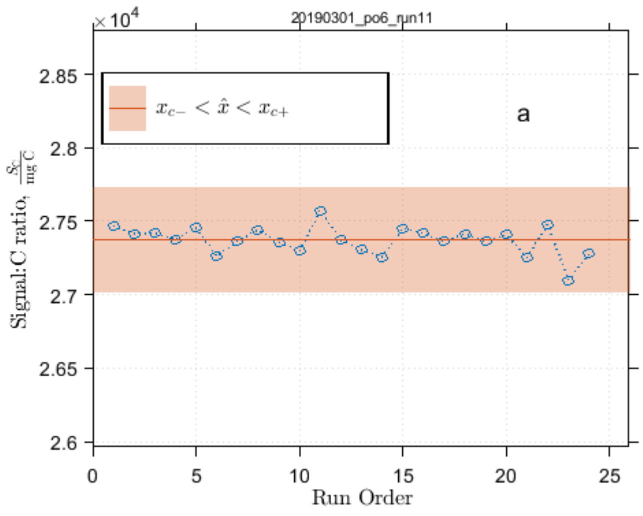
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



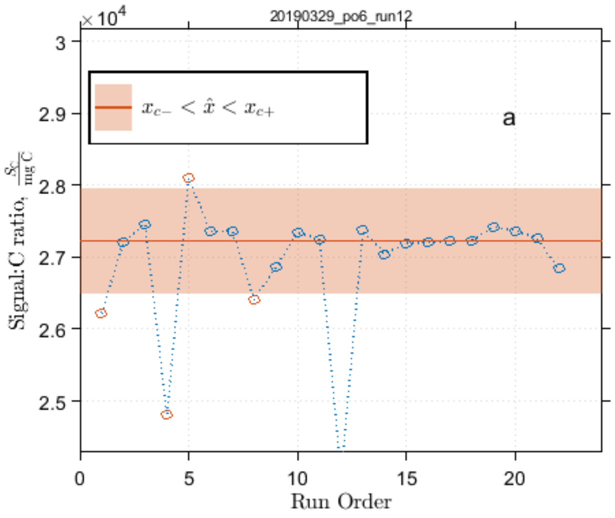
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



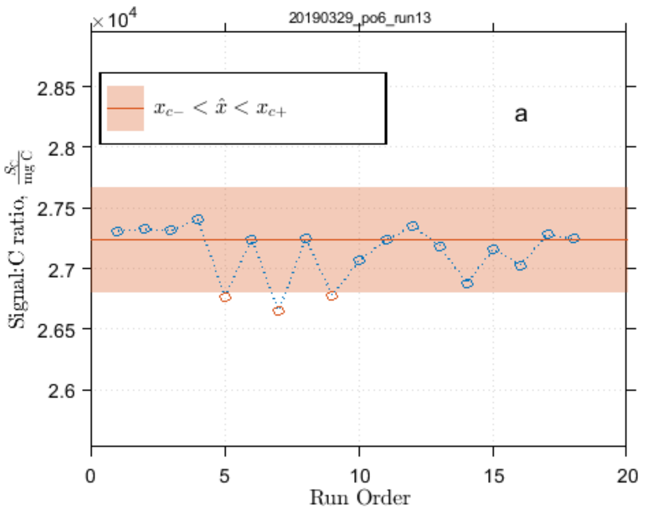
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



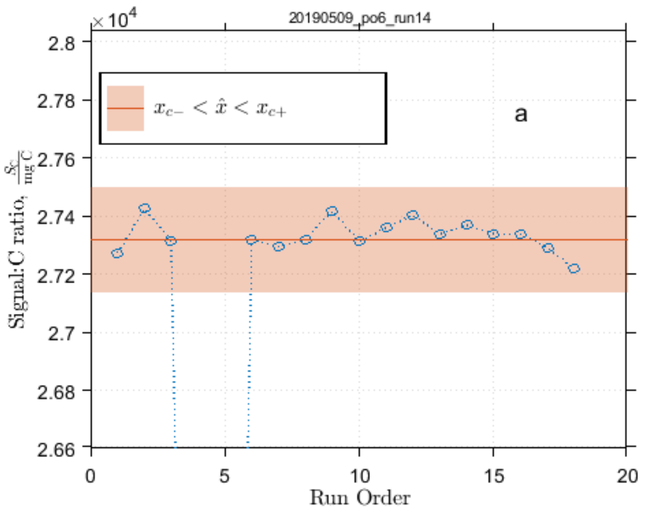
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



Plot Carbon Calibration curve

```
        % C regression statistics

           X =[ones(size(x,1),1) x];
           [b,bint,r,rint,stats] = regress(y,X);
```

Linear regression of sulfCarb as a f(x) of C signal -- post QA

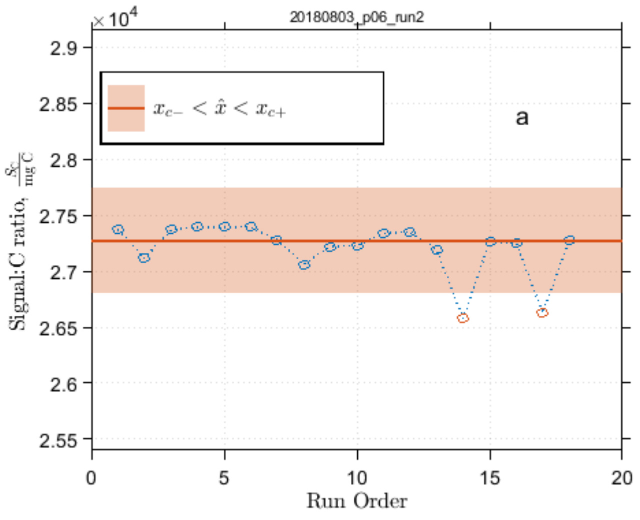
```
           xx = union(0,x);  % linear fit evaluation range

           carbModel = flipud(b)'; % polyfit(x,y,1);     %  Carbon mass vs C signal linear fit
           %carbModel(2) = 0; % substraction of intercept
           z = polyval(carbModel,xx);
```

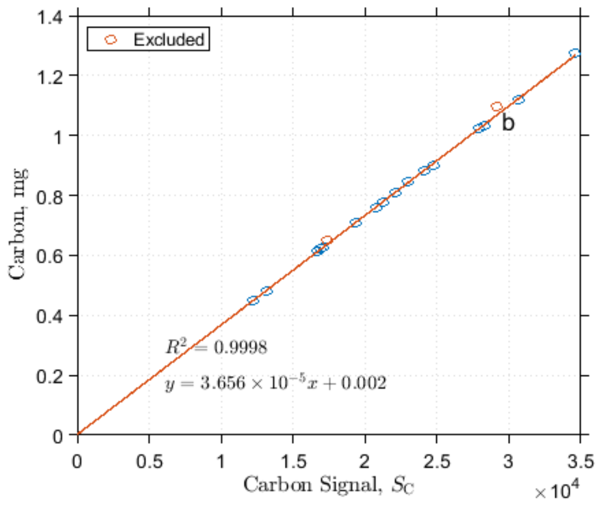
```
           subplot(2,1,2);

            L33 = plot(x,y,'o');
```

```
         L33.Color = newColors(1,:);

            grid on
            hold on

             L4 = plot(sulfCarbAreas(goodCalRuns == 0),sulfCarbWeight(goodCalRuns == 0),'o');

             if ~isempty(L4)
                L4.Color = newColors(2,:);
             end

             if sum(badCalRuns) > 0
                 l0 = legend(L4,'Excluded','AutoUpdate','off');
                 l0.Location = 'northwest';
             end

            aa = get(gca,'xlim');
            bb = get(gca,'ylim');

           L34 = plot(xx,z ,'-');
           L34.Color = newColors(2,:);

            yL2 = ylabel('Carbon, mg');
            yL2.Interpreter = 'Latex';

            xL2 = xlabel('Carbon Signal, $S_{\rm C}$');
            xL2.Interpreter = 'Latex';

            % Display R^2 & regression eq on cal curve plot

        tt01 = text(0.4 * aa(2),0.22 * bb(2),...
            ['$R^{2} = ' num2str(stats(1),4) '$'],...
            'fontsize',16,...
            'Interpreter','latex');
        tt01.Units = 'Inches';
        tt01.Position(1) = 1;
        tt01.HorizontalAlignment = 'left';

        newStr = regexp(num2str(b(2)),'e-','split');

        m = str2double(newStr{1});
        m = round(m,3);

        newStr{1} = num2str(m);

        regString = ['$ y =' newStr{1} '\times 10^{-' newStr{2}(2) '}x +'  num2str(b(1),2) '$'];

        tt02 = text(1.7,0.17,...
            regString,...
            'fontsize',16,...
            'Interpreter','latex');
        tt02.Units = 'Inches';
        tt02.Position(1) = 1.0;
        tt02.HorizontalAlignment = 'left';

        h2 = handle(gca);
        h2.LineWidth = LineWidths;
        h2.TickDir = 'out';
        h2.TickLength = h2.TickLength .* 1.5;


        tt1 = text(h2.XLim(2) * 0.8,h2.YLim(1) + range(h2.YLim * 0.75),'b');
        tt1.Units = 'Inches';
        %tt1.Position = [3.3571    2.5904         0];
        tt1.FontSize = 20;

    % Plot position adjustment

    %h2.Units = 'Inches';

    h2.Position = h2.Position .* [1.25 1.4 0.95 1];

    % Carbon

    sampleCarbAreas = carbonAreas(sampleDex) - sampleCarbBlank; % Sample Carbon areas blank corrected (minus tin square foil)
```

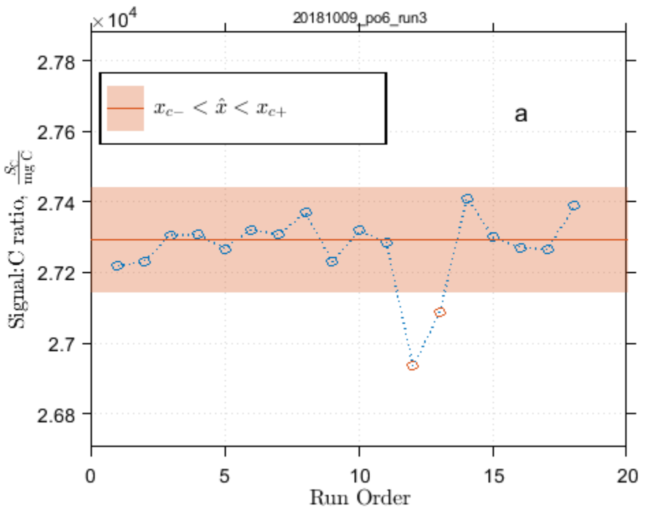
```
%sampleCarbWeight = polyval(carbModel,sampleCarbAreas);  % calculate sample Carbon in mg

% substracting the intercept of cal curve

sampleCarbWeight = polyval(carbModel,sampleCarbAreas) - carbModel(2);  % calculate sample Carbon in mg
```

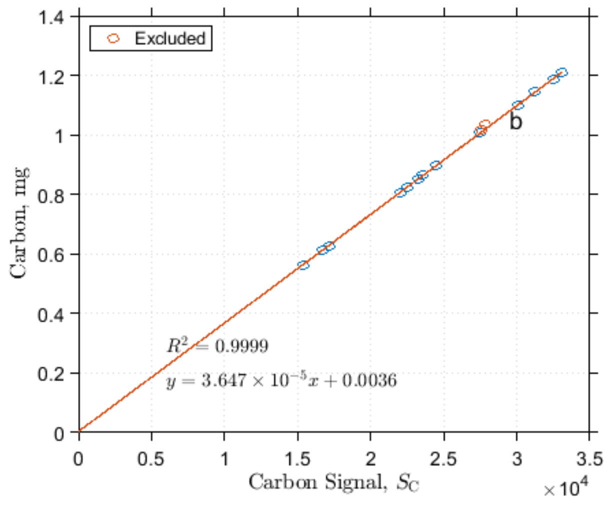
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



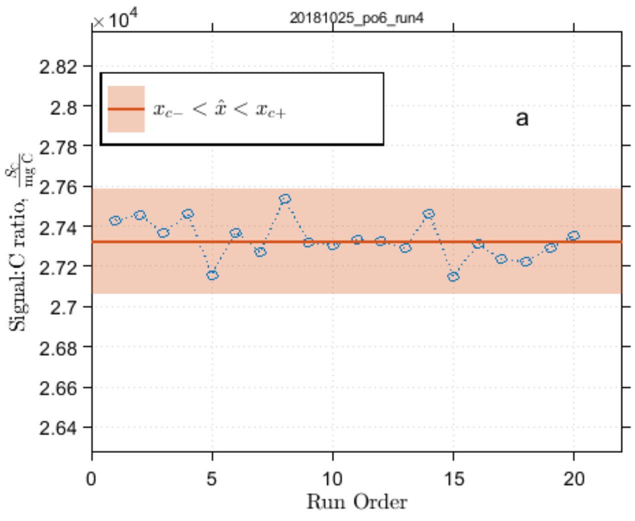


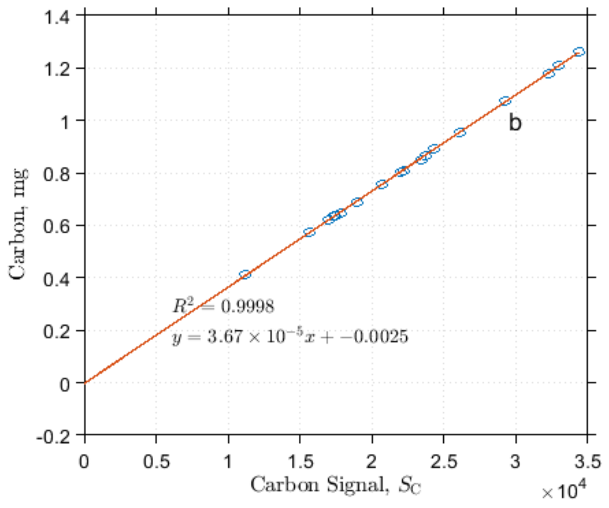
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



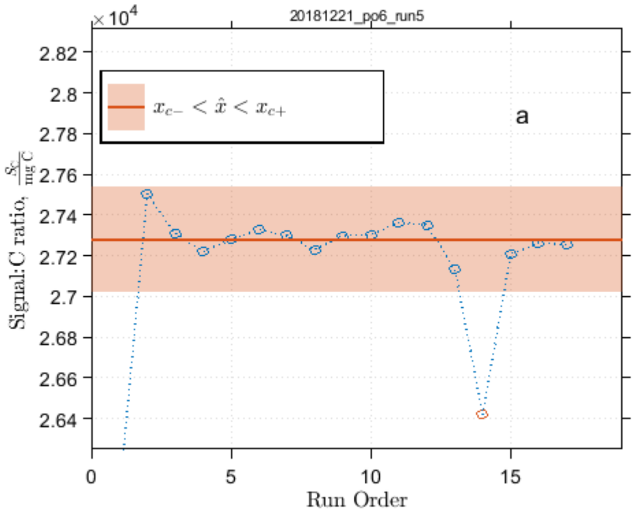


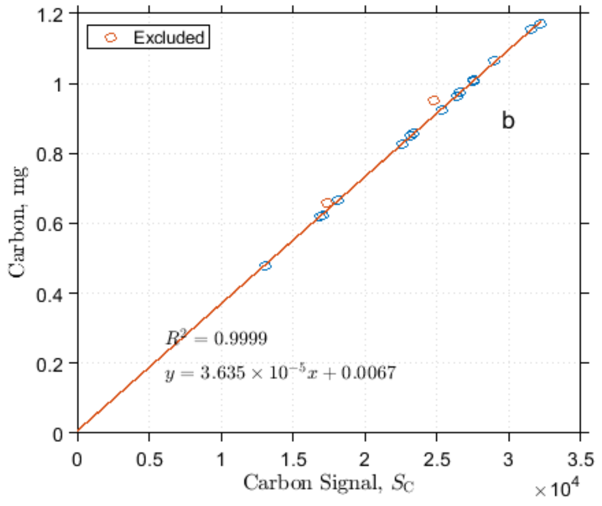
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



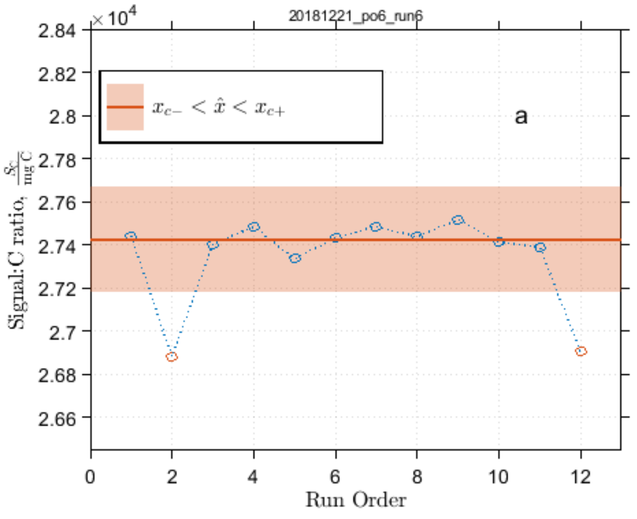


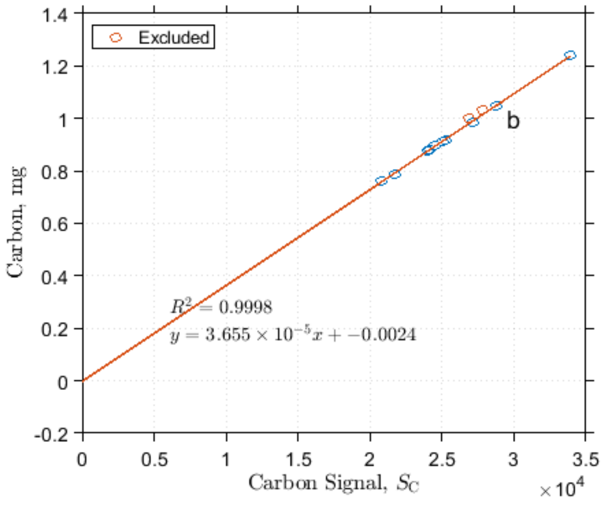
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



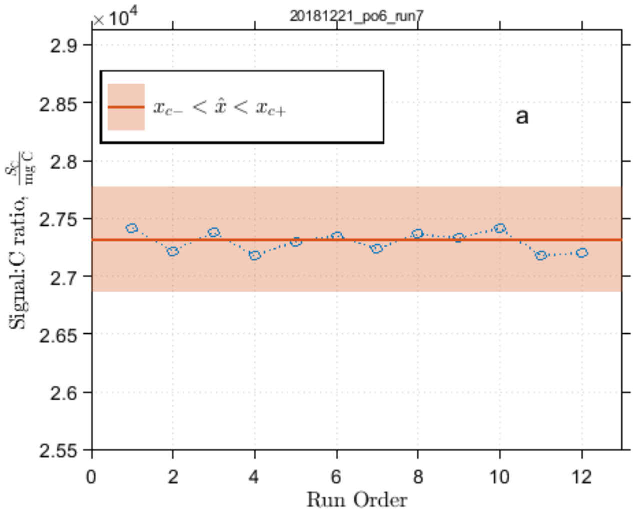


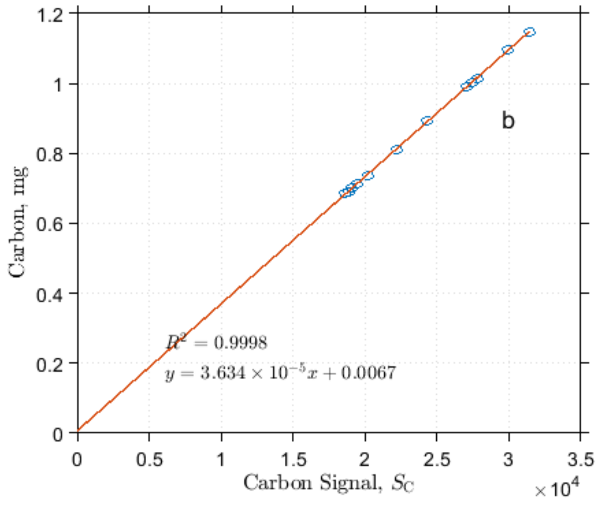
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



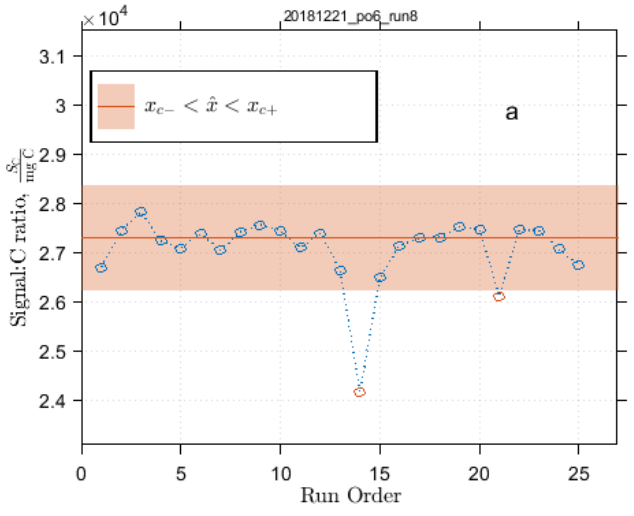


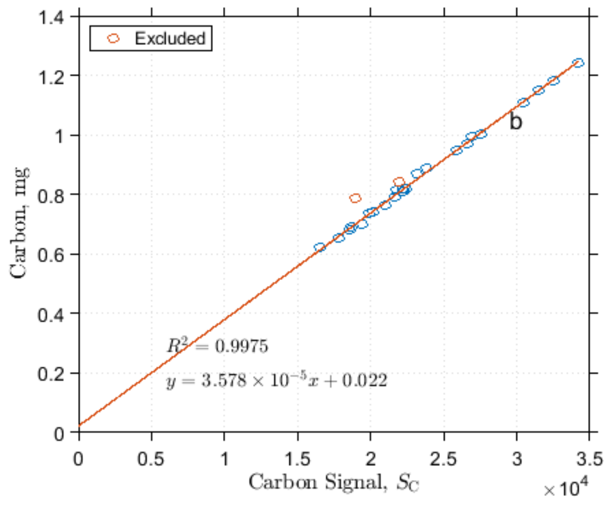
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



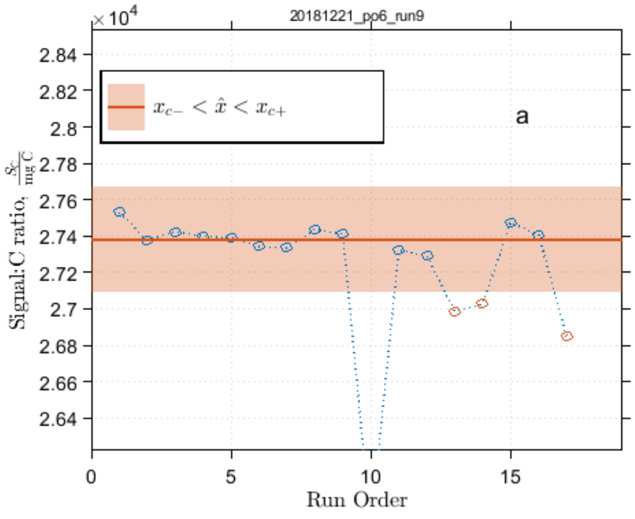


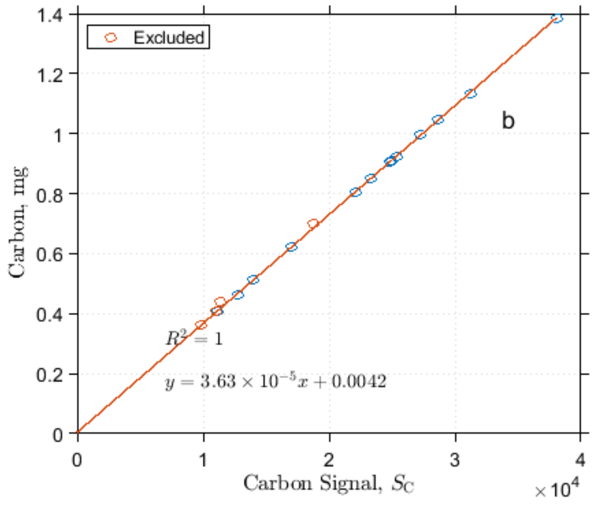
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



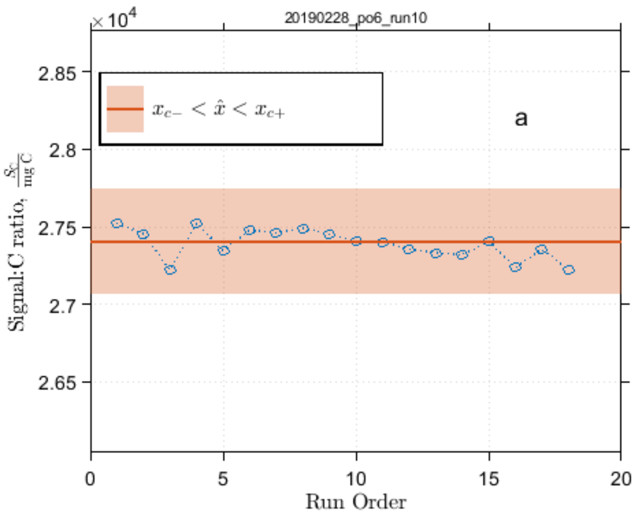


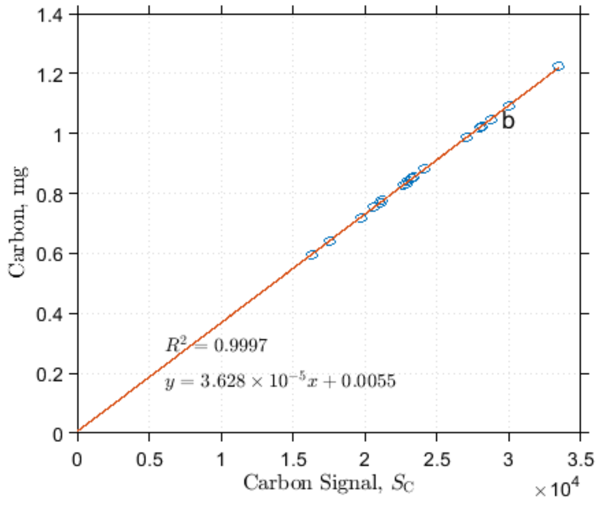
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



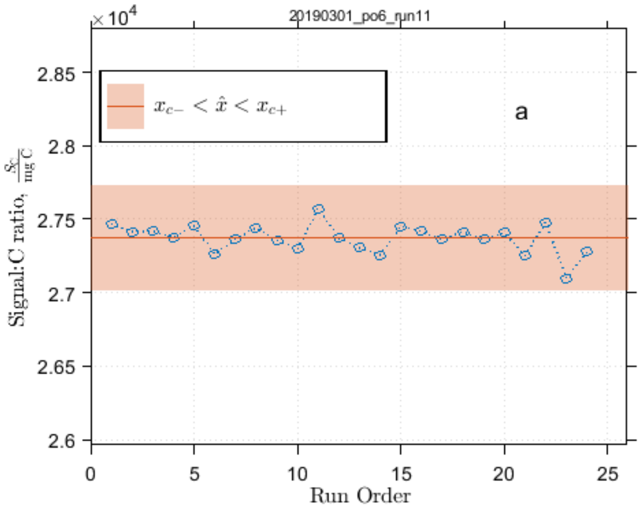


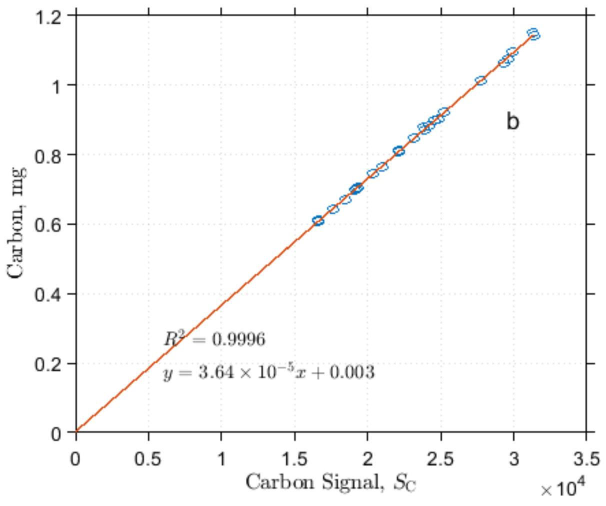
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



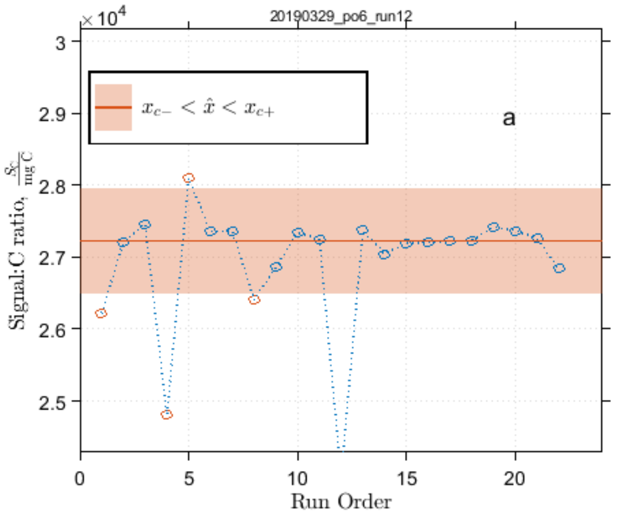


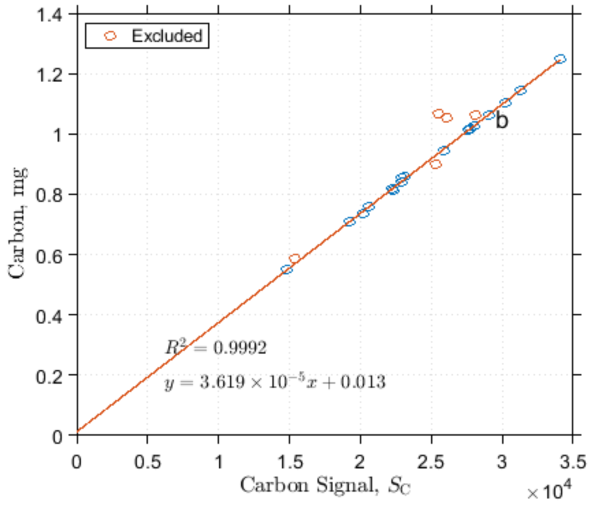
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



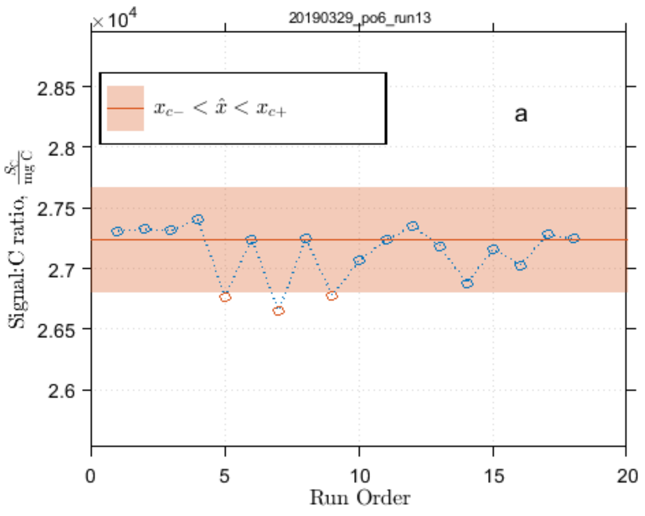


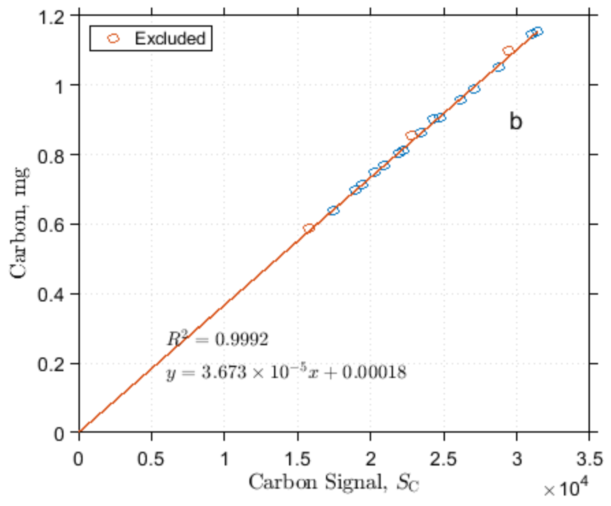
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



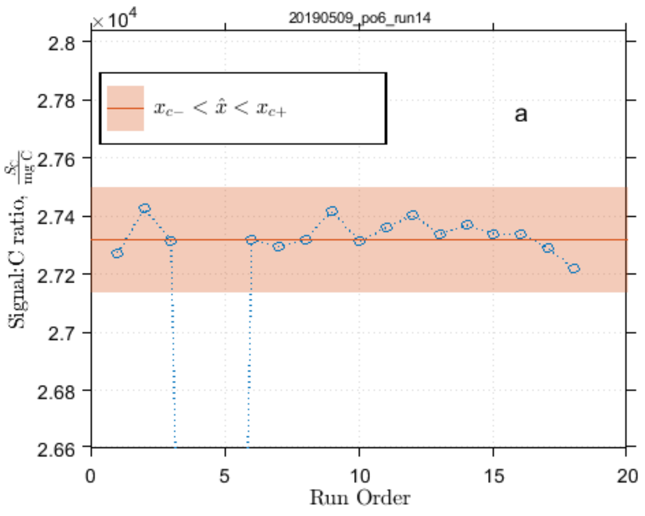


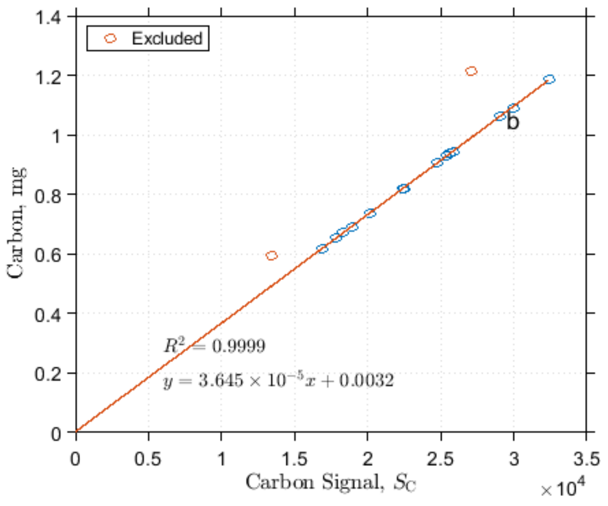
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```





```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```





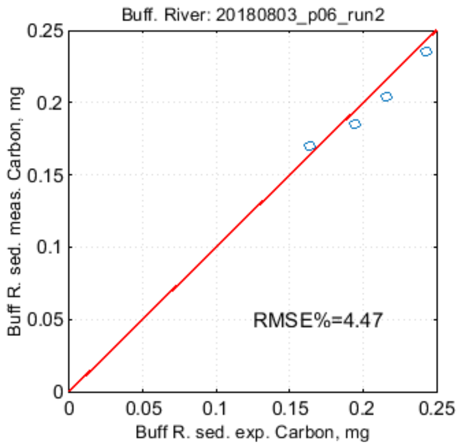
Data report of carbon and nitrogen per filter (NOT normalized by volume yet)

QA/QC with Buffallo River NIST reference material

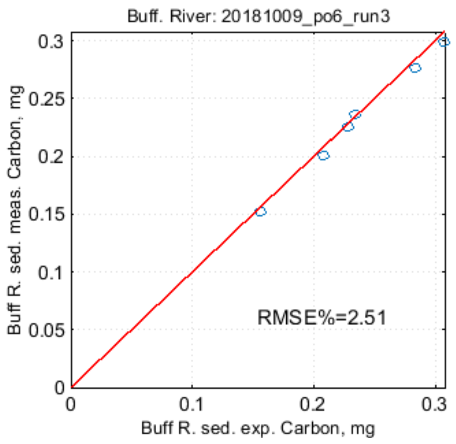
```
        % Carbon percent error based on Buffallo River runs
            % NIST certified C % content in Buff R. sed. is 3.351%

        %Buffalo River Sediment Ref stds

        buffRivMeasCarbon = polyval(carbModel,carbonAreas(buffRivDex) - calCarbBlank) - carbModel(2);  % measured C in Buff stds

        buffRivExpCarbon = 3.351/100 * weights(buffRivDex);        % expected C in Buff stds

    % tile figures

        if k == 1

            screenSize = get(0,'ScreenSize');

            plotWidth = 350;
            plotHeight = 325;
            nCols = floor(screenSize(3)/plotWidth);
            nRows = floor(screenSize(4)/plotHeight);

        end

        xPos = mod(k - 1,nCols) * plotWidth;
        if k > nCols
            yPos = (screenSize(4) - (67 + ceil(k/nCols) * plotHeight));
        else
            yPos = (screenSize(4) - (ceil(k/nCols) * plotHeight));
        end
        %test(k,1:3) = [k mod(k - 1,nCols) ceil(k/nCols) ];

         f = figure('Position',[xPos yPos plotWidth plotHeight]);
```

```
        plot(buffRivExpCarbon,buffRivMeasCarbon,'o')

            grid on
            hold on

           a=max(get(gca,'xlim'));
             set(gca,'xlim',[0 a]);
             set(gca,'ylim',[0 a]);
            line([0 a],[0 a],'color','r')

             xlabel('Buff R. sed. exp. Carbon, mg','fontsize',15)
             ylabel('Buff R. sed. meas. Carbon, mg','fontsize',15)

             error = errperf(buffRivExpCarbon,buffRivMeasCarbon,'rmspe');

             text(a * 0.5, a * 0.2,['RMSE%=' num2str(error,3)],'fontsize',17)

             ti1 = title(['Buff. River: ' runFileStr{1}],...
                 'fontsize',15);
             ti1.Interpreter = 'none';
             ti1.FontWeight = 'normal';
```

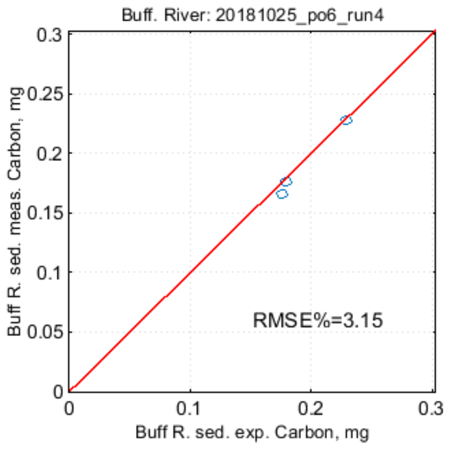
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



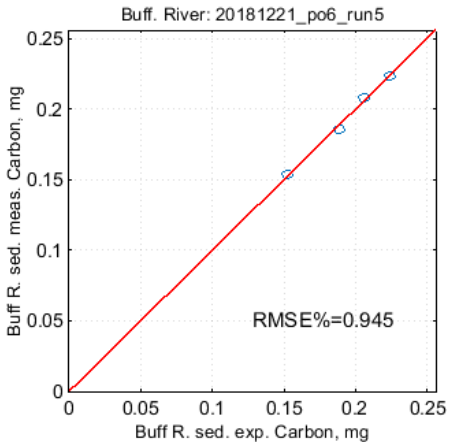
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



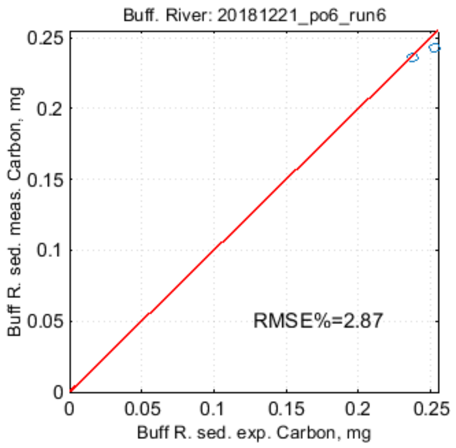
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



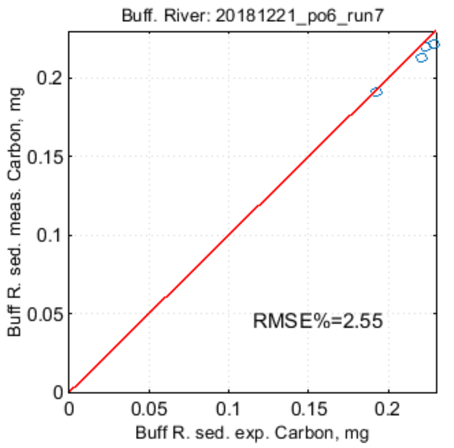
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



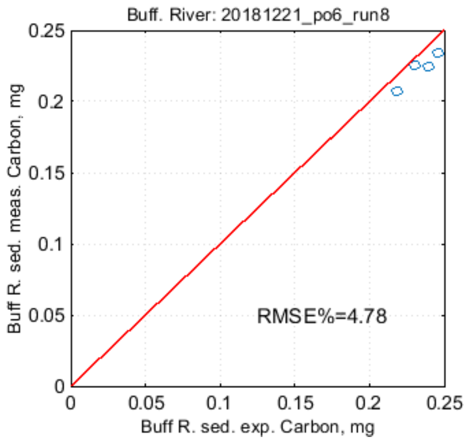
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



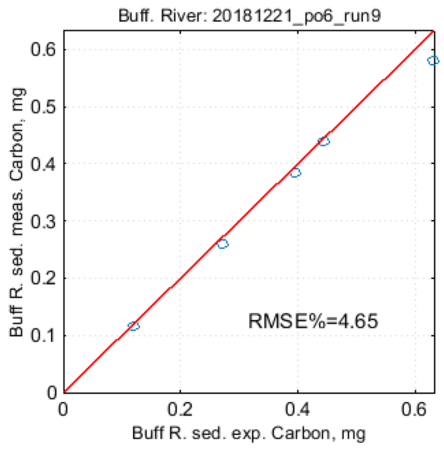
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



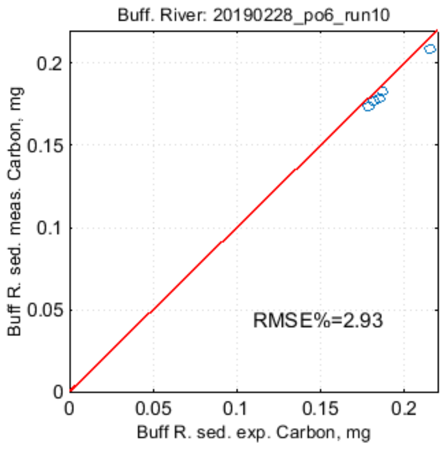
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



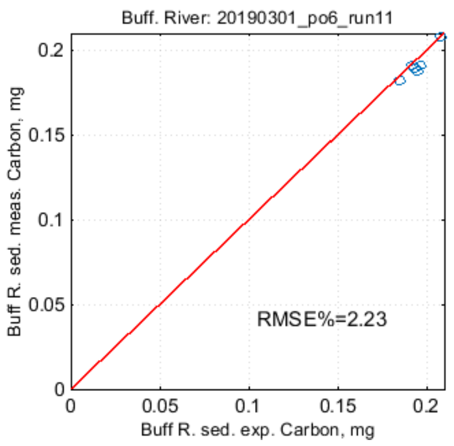
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



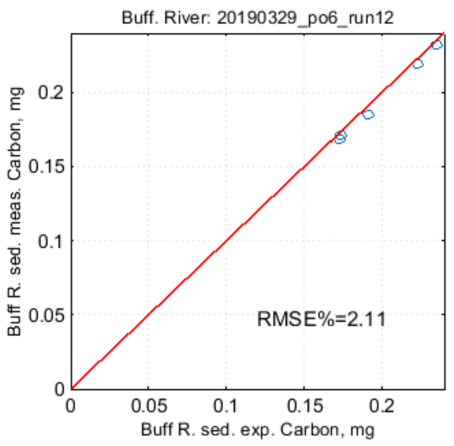
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



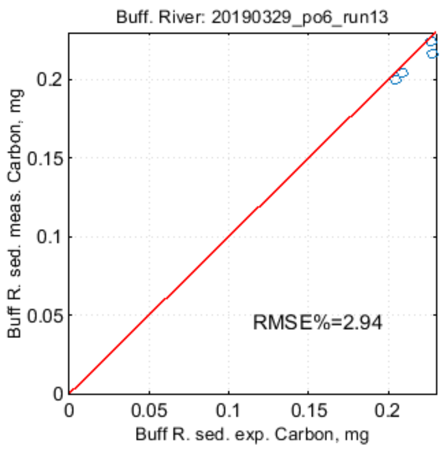
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



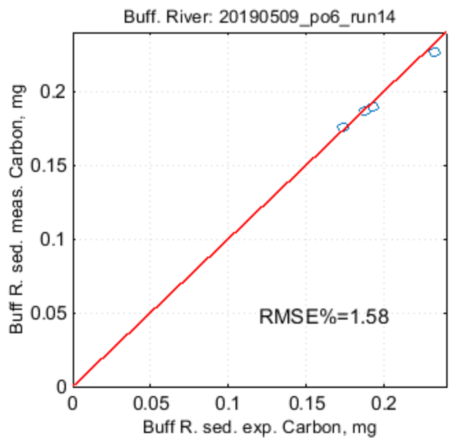
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



```
        %% Export C concentration data to a file along with sample ID

        sampleID = str2double(runNames(sampleDex));

    % if sampleID are not being coverted to numbers try pass them directly

    if sum(isnan(sampleID)) == numel(sampleID)
        sampleID = cell2num(runNames(sampleDex));

    elseif any(isnan(sampleID))

        a = runNames(sampleDex);

        sampleID(isnan(sampleID)) = cell2num(a(isnan(sampleID)));
    end

if k == 22
    stop
end
        runNumber = repmat(k,numel(sampleID),1);

        M = [runNumber sampleID sampleCarbWeight];

                if k == 1
                        dlmwrite('P06carbonWeigths.txt',M,...
                        'delimiter',' ','roffset',0,'precision',9)

                else

                        dlmwrite('P06carbonWeigths.txt',M,'-append',...
                        'delimiter',' ','roffset',0,'precision', 9);
            end
```

```
        %% Export calibration curve and other performance metrics to file

    runNumberStr = sprintf('runNumber_%d',k);

    runData.(runNumberStr).carbModel = carbModel;
    runData.(runNumberStr).buffRMSE = error;
    runData.(runNumberStr).buffData = [buffRivExpCarbon buffRivMeasCarbon];
    runData.(runNumberStr).calCarbBlankReps = calCarbBlankReps;
    runData.(runNumberStr).sampleCarbBlankReps = sampleCarbBlankReps;

    if k == length(runFiles)
        save('P06runDataStrct.mat','runData')
    end
```

```
end
```

Export figures to file

```
  ag = findobj; % all graphical objects

  nf = ag(2).Number; %max(ag(ag==fix(ag))); % maximal integer value is number of figures


if strcmpi(exportFigs,'yes')

     %cd('/Users/jchavesc/Documents/CVO/Biogeochem/POC/Runs/Figures')
     %cd('/Users/jchavesc/Documents/CVO/Workshops/POC_mfiles')
     %cd(['~/jchavesc/Documents/CVO/Workshops/POC_mfiles'])  % Re direct to where

    cd('~/Documents/CVO/Workshops/Writing/POCLatexVersion/Figures')

       for i = 1:nf

           print(i,'-dpdf',['Fig' num2str(i) 'CubeRun' rundID '.pdf'])
           print(i,'-dpng','calCurveFigv2_latest.png','-r600')

       end

end
```

```
      disp('Process completed')
      toc
```

```
Process completed
Elapsed time is 253.391700 seconds.
```

Published with MATLAB® R2019b

Contents

```
% Process CHN runs from Elementar Cube analyzer
```

```
clearvars
close all
tic

% Runs Info

cruise = 'P06_2017';   %enter here cruise or campaign code
```

Settings

```
calQA = 'yes'; % assess quality of cal runs and remove suspect ones

exportFigs = 'no';

LineWidths = 1.6;

newColors = get(groot,'DefaultAxesColorOrder');

scaledMADs = 3.5;  % number os scaled Median Absolute deviations for...
                   % calibration run to be classified as an outlier

if ispc
    homeStr = 'C:/Users/jchavesc';
else
    homeStr = '~';
end
```

Identify all files with runs

```
path2Data = [homeStr '/Documents/CVO/Biogeochem/POC/P06_Runs/'];

cd(path2Data)

if ispc
    delete('._2*') % removes hidden files created by Excel
else
    system('rm ._2*') % removes hidden files created by Excel
end

runFiles = dir('*.xlsx');
```

```
rm: cannot remove '._2*': No such file or directory

ans =

     1
```

Process each run file

```
for k = 1:length(runFiles)
```

Data info

```
        rundID = num2str(k); % run ID (temp) *****

        runFileStr = strsplit(lower(runFiles(k).name),'_fac'); % to tag plots
```

Open Data File that contains analysis report from CHN instrument

your data is located

```
        sheetStr = 'samples'; % sheet tab name
        % open data and put into numeric, text, and 'raw' outputs

        [num,txt,raw] = xlsread(runFiles(k).name,sheetStr);

        % use data from 'raw' import from XLS file
        % and remove header row

        data = raw(2:end,:);

        % extract 1st line of headers
        headers = txt(1,1:end);
        headers = string(headers); % convert to a vector of strings

        % Column for run 'Name'

        namesCol = headers == "Name";
```

Find index for each type of run

```
        runLength = length(data);

        % Preallocate vectors for run class indeces

        sulfDex = zeros(size(runLength));   % index of sulfanilamide standards
        airDex  = zeros(size(runLength));   % index of air blanks
        tinCalDex = zeros(size(runLength));   % index tin boats blanks
                                        % (i.e.  for running weighed standards)
        tinSampDex = zeros(size(runLength));   % index tin circle blanks (i.e. for running filter with samples or filter blanks)
        buffRivDex = zeros(size(runLength));   % index Buffalo River Sediment reference runs
        sampleDex =  zeros(size(runLength));   % index of sample runs
        sampleSeq =  zeros(size(runLength));   % Sample ID

        % Preallocate for tin blanks means from all runs to fill runs
        % without blank runs

        if k == 1
            allNitroCalBlanks = zeros(k,1);
            allNitroSampBlanks = zeros(k,1);
        end

        % Column with names/ID entered for each run

        runNames = data(:,namesCol);

        for i = 1:runLength

            if isnumeric(runNames{i}) % if runNames{i} convert directly to number is a sample ID

                 sampleDex(i) = 1;
                 sampleSeq(i) = runNames{i};

            else

                 runStr = runNames{i}; % string entered for individual run

                 % Cleanout runStr for spaces and make it all lowercase

                 runStr = lower(runStr(~isspace(runStr)));

                 % clean NaNs

                 if isnan(runStr)
                     runStr = 'empty';
                 end

                 % sulfanilamide
                 sulfDex(i) = any(regexpi(runStr,'sulfa'));  % sulfanilamide standards i.e., any runStr containing 'sulfa...'
                 sulfDex = logical(sulfDex);

                 % air blanks
                 airDex(i) = any(regexpi(runStr,'air'));  % air blanks i.e., any runStr containing 'air...'
                 airDex = logical(airDex);

                 % acidified filter blanks
                %  acidFiltDex(i) = any(regexpi(runStr,'acid'));  % sulfanilamide standards i.e., any runStr containing 'sulfa...'
                %  acidFiltDex = logical(acidFiltDex);

                 % Tin blanks (boats and sheets)

                     if any(regexpi(runStr,'tin'))

                         if any(regexpi(runStr,'35x35')) || any(regexpi(runStr,'30x30'))
                             tinSampDex(i) = 1;
                         else
                             tinCalDex(i) = 1;
                         end
                     end

                      tinSampDex = logical(tinSampDex);
                      tinCalDex = logical(tinCalDex);

                     % Sample runs numbers stored as string

                      if ~isnan(str2double(runStr))  % if run string converts to number

                          sampleDex(i) = 1;
```

```
                        sampleSeq(i) = str2double(runStr);

                    end

                   % Buffallo River NIST reference

                   buffRivDex(i) = any(regexpi(runStr,'buff'));  % Buffallo River reference i.e., any runStr containing 'buff...'
                   buffRivDex = logical(buffRivDex);

          end
                   sampleDex = logical(sampleDex);
      end
```

Instrument signals for each element

Nitrogen: Nitrogen signal data.Area1

```
          % All Nitrogen areas

           area1Col = headers == "Area1";

           nitroAreas = data(:,area1Col);           %  Nitroge signals (i.e. Area1)

           nitroAreas = cell2num(nitroAreas);

          % calibration (tin boats) blanks

           calNitroBlankReps = nitroAreas(tinCalDex);  %  N blanks for calibration runs

           calNitroBlank = nanmean(calNitroBlankReps);      %  Mean N blanks for calibration runs ONLY

           allNitroCalBlanks(i) =  calNitroBlank;

            % sample tin circle blanks

           sampleNitroBlankReps = nitroAreas(tinSampDex);  %  N tin blanks for sample runs

           sampleNitroBlank = mean(sampleNitroBlankReps);    %  Mean N tin blanks for sample runs

           allNitroSampBlanks(i) = nanmean(sampleNitroBlankReps);

          % Acidified (Method) filter blanks

        %     filtNitroBlankReps = nitroAreas(acidFiltDex);  % filter blanks
        %
        %     filtNitroBlank = mean(filtNitroBlankReps);
```

Run CARBON calibration curves

```
            % Nitrogen -- Amount of C in mg for sulfanilamide runs

            weightCol = headers == "WeightVol";

            weights = data(:,weightCol);

            weights = cell2num(weights);

%            sulfCarbWeight   = (41.8/100).* weights(sulfDex); % mg C per sulfanilamide run

            sulfNitroWeight   = (16.3/100).* weights(sulfDex); % mg N per sulfanilamide run

            if ~isnan(calNitroBlank)

                sulfNitroAreas  = nitroAreas(sulfDex) - calNitroBlank;  % Sulfanilamide areas, blank corrected

            else
                % Use mean cal blanks from previous runs if no blanks
                % available

               sulfNitroAreas  = nitroAreas(sulfDex) - nanmean(allNitroCalBlanks(allNitroCalBlanks > 0));  % Sulfanilamide areas, blank corrected

            end

            if isnan(calNitroBlank) || isnan(sampleNitroBlank)
                warning('calNitroBlank and/or calNitroBlank were NaN; mean of previous runs used')
            end

         x = sulfNitroAreas;
         y = sulfNitroWeight;
```

```
Warning: calNitroBlank and/or calNitroBlank were NaN; mean of previous runs used


Warning: calNitroBlank and/or calNitroBlank were NaN; mean of previous runs used
```

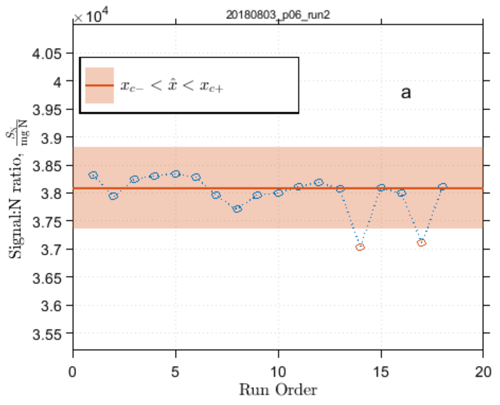
Calibration run QA

```
        if strcmpi(calQA,'yes')


            areaNitroRatio = x./y;  % area to nitroon ratio in sulfanilamide stds
            medianNitroRatio = nanmedian(areaNitroRatio);
            stdNitroRatio = std(areaNitroRatio);
```

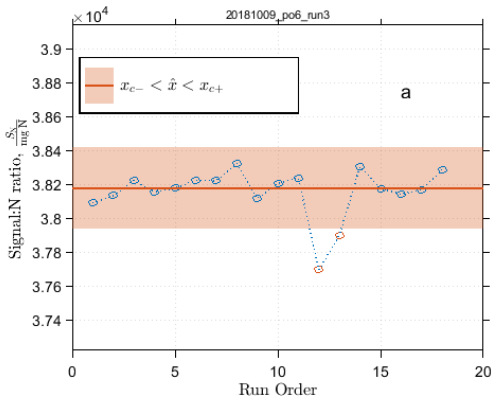
```
absNitroResid = abs(areaNitroRatio - medianNitroRatio);
maxNitroResid = max(absNitroResid);
MADNitroRatio = mad(areaNitroRatio,1);
modZScores = 0.6745 * ((areaNitroRatio - medianNitroRatio) ./ MADNitroRatio);

% cal QA plot
% Tweaks so that figures renders the same in Mac & Linux

screens = handle(0); %
mainScreen = screens.MonitorPositions(1,3:4);

if ismac
    h0i = figure('Position',[mainScreen(1) * 1.06 378 364 636]);
elseif isunix
    h0i = figure('Position',[mainScreen(1) * 1.06 209 500 900]);
    h0i.Renderer = 'OpenGL';
end

subplot(2,1,1)

L1 = plot(areaNitroRatio,'o:');
hold on

% axes properties

h1 = handle(gca);
h1.XLim = [0 ceil(numel(areaNitroRatio) * 1.06)];
h1.LineWidth = LineWidths;
 grid on

h1.TickDir = 'out';
h1.TickLength = h1.TickLength .* 1.5;

L2 = hline(medianNitroRatio);
L2.LineStyle = '-';
L2.Color = newColors(2,:);

% An outlier is a value that is more than three SCALED median absolute ...
% deviations (MAD) away from the median:
%
% For a random variable vector A made up of N scalar observations,
% the median absolute deviation (MAD) is defined as

%     MAD' = median(abs(A-median(A)));
%
%  The scaled MAD is defined as c * MAD
%  where c = 1.4826 and is given by:
% -1/(sqrt(2)*erfcinv(3/2)).
%
%  [***NOTE***] In Matlab mad(x) gives the MEAN absolute deviation from
%  the media & while in R the function mad(x) gives the MEDIAN absolute
%  deviation from the median. To get the scaled MAD in Matlab use the
%  syntax mad(x,1)

% Graphical implementation of the above with the threshold values

X = [h1.XLim fliplr(h1.XLim)];

c = -1/(sqrt(2)*erfcinv(3/2)); %

M_i = scaledMADs * c * MADNitroRatio;

    y1 = medianNitroRatio - M_i;
    y2 = fliplr(medianNitroRatio + M_i);
Y = [y1 y1 y2 y2];
L3 = patch(X,Y,L2.Color);
L3.FaceAlpha = 0.3;
L3.EdgeColor = 'none';

% Y Lims

f = 4;
h1.YLim = [(medianNitroRatio - (M_i * f))  (medianNitroRatio + (M_i * f))];

%    L1 = plot(areaNitroRatio,'o:');

%hline([meanNitroRatio meanNitroRatio-stdNitroRatio  meanNitroRatio+stdNitroRatio],{'b', 'r', 'r'},{'mean+/-s.d.','',''})
xLab1 = xlabel('Run Order');
xLab1.Interpreter = 'latex';
yLab1 = ylabel('Signal:N ratio, $\frac{S_{\rm N}}{{\rm mg\, N}}$');
yLab1.Interpreter = 'latex';
%title(['C calibration Q/A ' rundID  ' run'])

% ID oulier Sulfanilamide runs (Mean )areaNitroRatio

goodCalRuns = abs(modZScores) < scaledMADs; % ~isoutlier(areaNitroRatio,'median',3);  % index of cal runs to keep
badCalRuns = abs(modZScores) > scaledMADs; % isoutlier(areaNitroRatio,'median',3);    % index of cal runs to keep

xx = 1:numel(areaNitroRatio);

L11 = plot(xx(badCalRuns),areaNitroRatio(badCalRuns),'o');
if ~isempty(L11)
    L11.Color = newColors(2,:);
end

x = x(goodCalRuns);
y = y(goodCalRuns);
```

```
            % Legends

            LegL3 = patch(h1.XLim(2) * [0.03 0.03 0.1 0.1],h1.YLim(1) + (range(h1.YLim) * [0.76 0.87 0.87 0.76]),L2.Color);
            LegL3.FaceAlpha = 0.3;
            LegL3.EdgeColor = 'none';
            LegL4 = plot([h1.XLim(2) h1.XLim(2)] .* [0.03 0.1],h1.YLim(1) + (repmat(range(h1.YLim),1,2) .* [0.815 0.815]));
            LegL4.LineWidth = LineWidths;
            LegL4.Color = L2.Color;
            ttL1 = text(h1.XLim(2) * 0.1167,h1.YLim(1) + (range(h1.YLim) * 0.815),'$x_{c-} < \hat{x} < x_{c+}$');
            ttL1.Interpreter = 'Latex';
            ttL1.FontSize = 18;

            tt01 = text(h1.XLim(2) * 0.8,h1.YLim(1) + range(h1.YLim * 0.8),'a');
            tt01.Units = 'Inches';
            tt01.Position = tt01.Position;
            tt01.FontSize = 20;

            % Box
            bx = patch(h1.XLim(2) * [0.018 0.018 0.55 0.55],h1.YLim(1) + (range(h1.YLim) * [0.73 0.9 0.9 0.73]),'w');
            bx.LineWidth = LineWidths;


        ti01 = title(runFileStr{1});
        ti01.FontWeight = 'normal';
        ti01.Interpreter = 'none';
        ti01.FontSize = 12;

        end
```

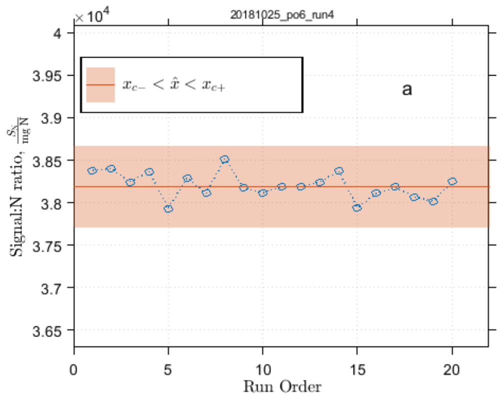
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



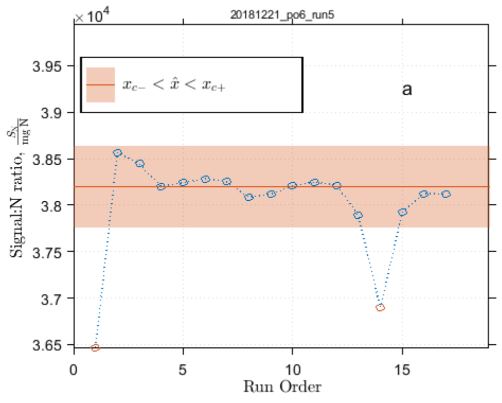
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



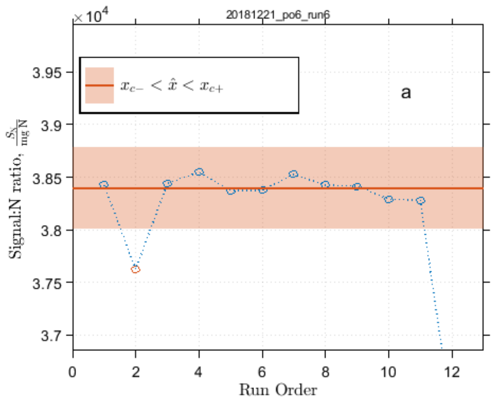
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



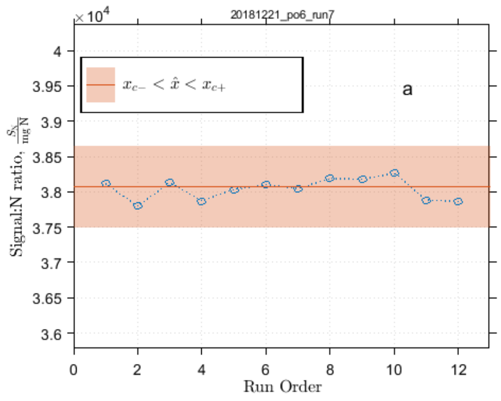
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



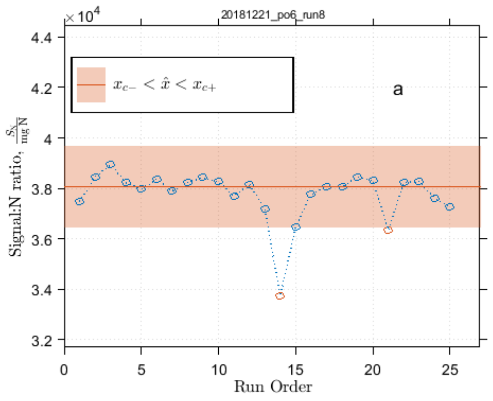
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



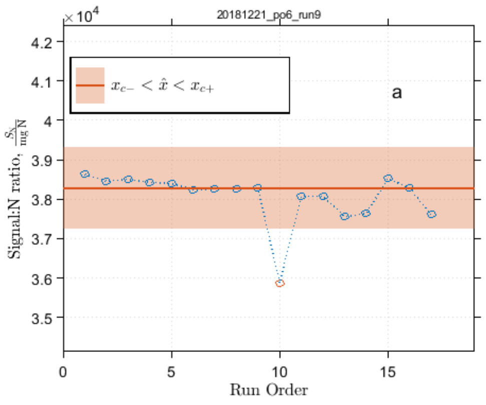
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



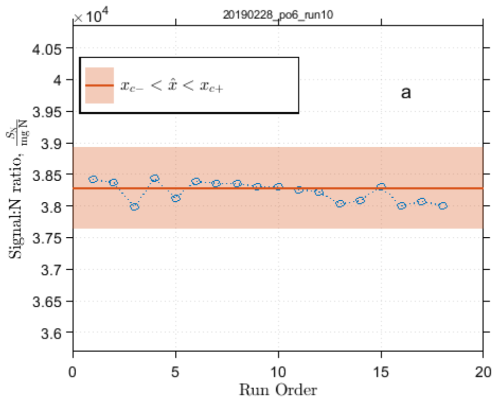
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



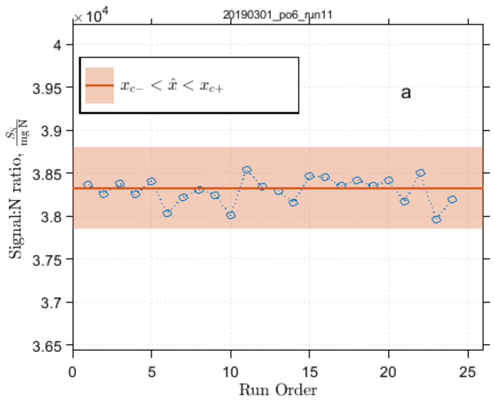
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



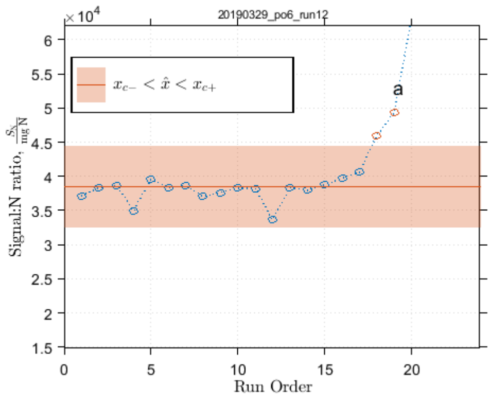
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



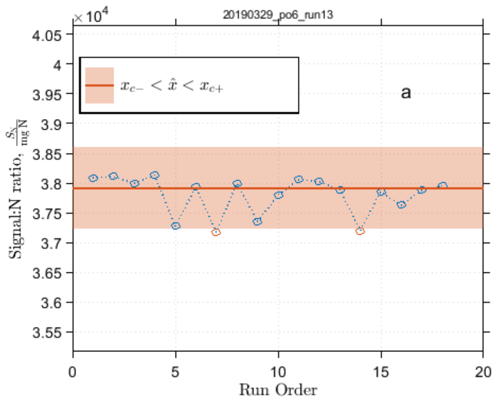
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



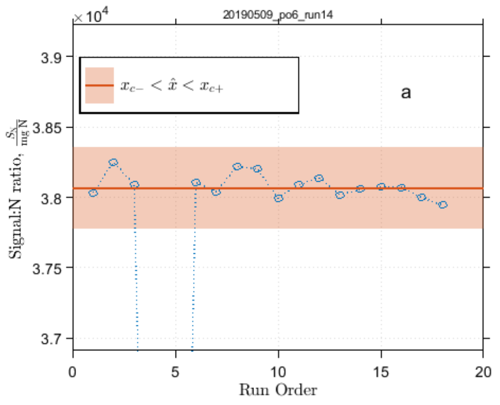
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



Plot Nitrogen Calibration curve

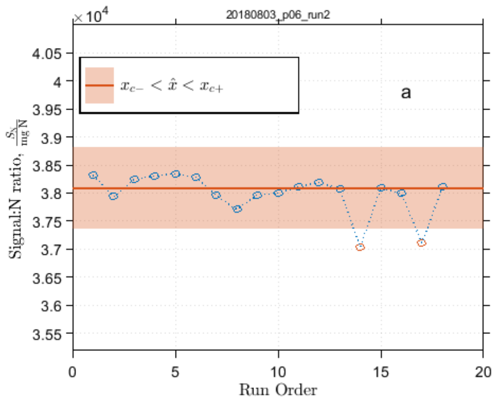
```
        % C regression statistics

           X =[ones(size(x,1),1) x];
           [b,bint,r,rint,stats] = regress(y,X);
```

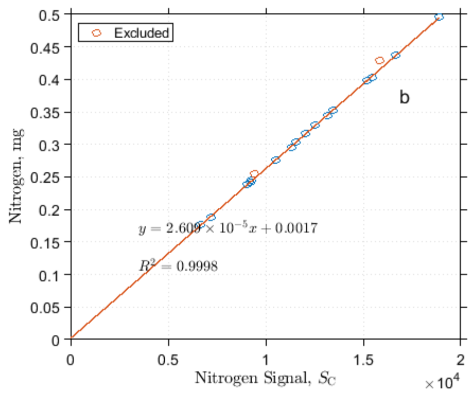
Linear regression of sulfNitro as a f(x) of C signal -- post QA

```
           xx = union(0,x);  % linear fit evaluation range

           nitroModel = flipud(b)'; % polyfit(x,y,1);     %  Nitrogen mass vs C signal linear fit
           %carbModel(2) = 0; % substraction of intercept
           z = polyval(nitroModel,xx);
```

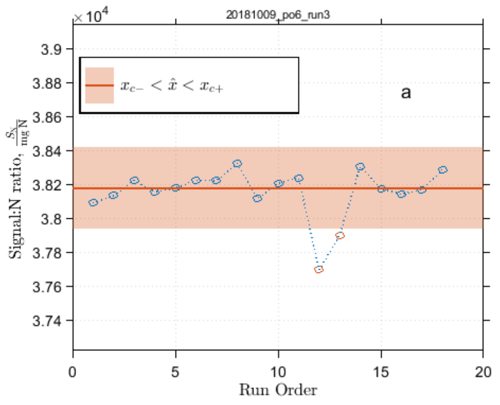
```
           subplot(2,1,2);

            L33 = plot(x,y,'o');
            L33.Color = newColors(1,:);

               grid on
               hold on

                L4 = plot(sulfNitroAreas(goodCalRuns == 0),sulfNitroWeight(goodCalRuns == 0),'o');

                if ~isempty(L4)
                   L4.Color = newColors(2,:);
                end

                if sum(badCalRuns) > 0
                    l0 = legend(L4,'Excluded','AutoUpdate','off');
                    l0.Location = 'northwest';
                end

               aa = get(gca,'xlim');
               bb = get(gca,'ylim');

              L34 = plot(xx,z ,'-');
              L34.Color = newColors(2,:);
               yL2 = ylabel('Nitrogen, mg');
               yL2.Interpreter = 'Latex';

               xL2 = xlabel('Nitrogen Signal, $S_{\rm C}$');
               xL2.Interpreter = 'Latex';

               % Display R^2 & regression eq on cal curve plot

           tt01 = text(0.4 * aa(2),0.22 * bb(2),...
               ['$R^{2} = ' num2str(stats(1),4) '$'],...
               'fontsize',16,...
               'Interpreter','latex');
           tt01.Units = 'Inches';
           tt01.Position(1) = 1;
           tt01.HorizontalAlignment = 'left';

           newStr = regexp(num2str(b(2)),'e-','split');

           m = str2double(newStr{1});
           m = round(m,3);

           newStr{1} = num2str(m);

           regString = ['$ y =' newStr{1} '\times 10^{-' newStr{2}(2) '}x +'  num2str(b(1),2) '$'];

           tt02 = text(1.7,0.17,...
               regString,...
               'fontsize',16,...
               'Interpreter','latex');
           tt02.Units = 'Inches';
           tt02.Position(1) = 1.0;
           tt02.HorizontalAlignment = 'left';

           h2 = handle(gca);
           h2.LineWidth = LineWidths;
           h2.TickDir = 'out';
           h2.TickLength = h2.TickLength .* 1.5;


           tt1 = text(h2.XLim(2) * 0.8,h2.YLim(1) + range(h2.YLim * 0.75),'b');
           tt1.Units = 'Inches';
           %tt1.Position = [3.3571    2.5904         0];
           tt1.FontSize = 20;

       % Plot position adjustment

       %h2.Units = 'Inches';
       h2.Position = h2.Position .* [1.25 1.4 0.95 1];
```

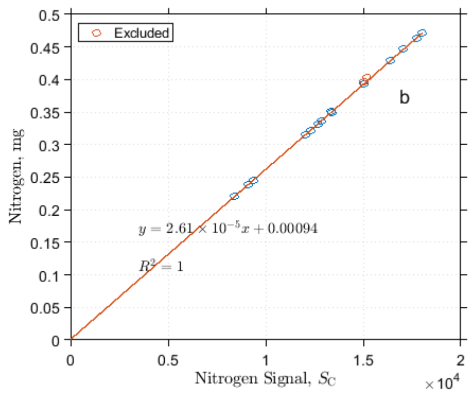
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



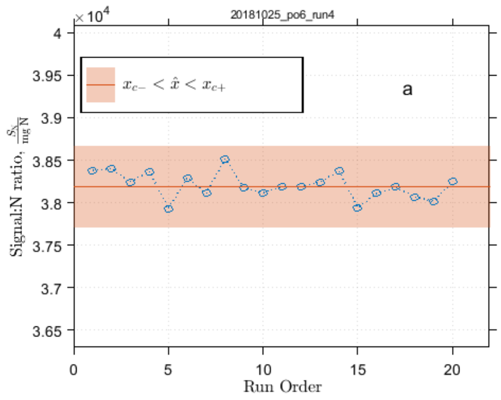


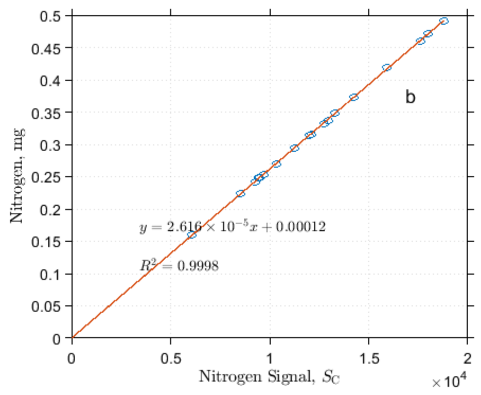
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



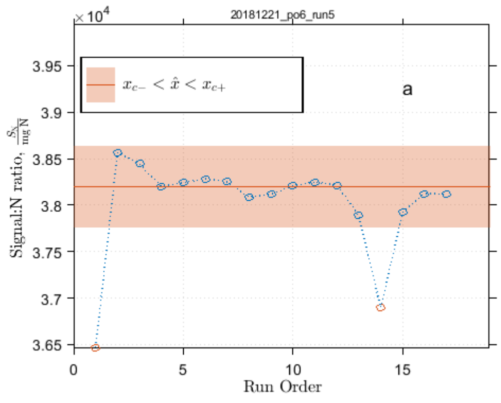


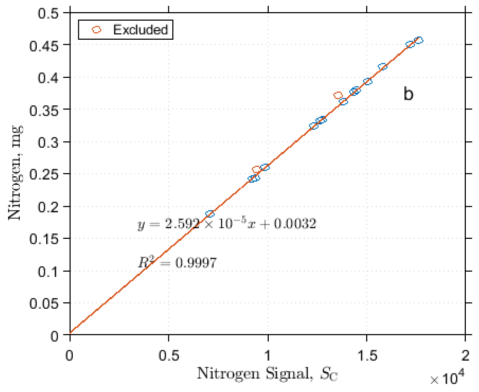
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



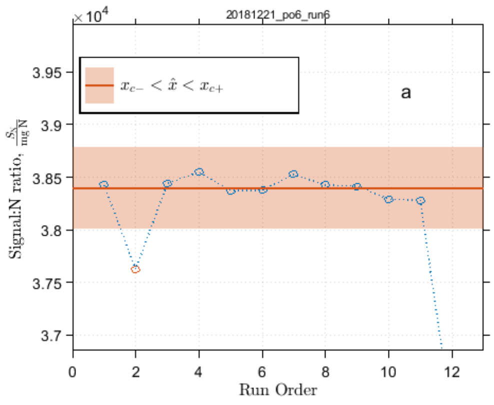


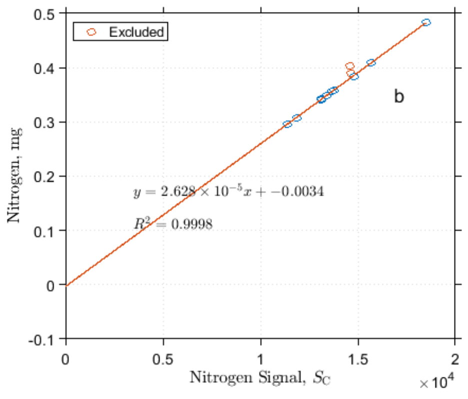
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



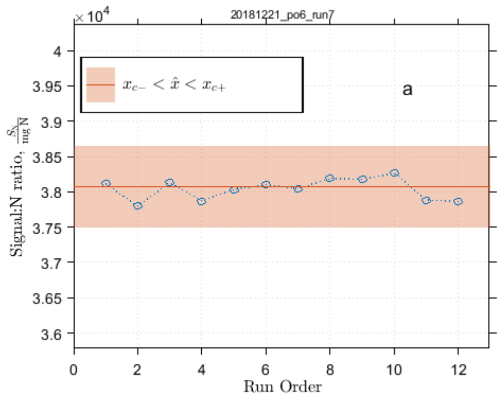


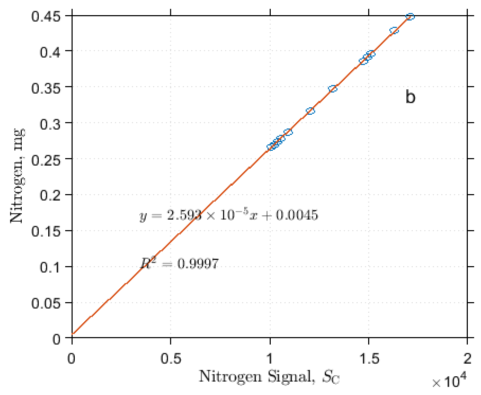
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



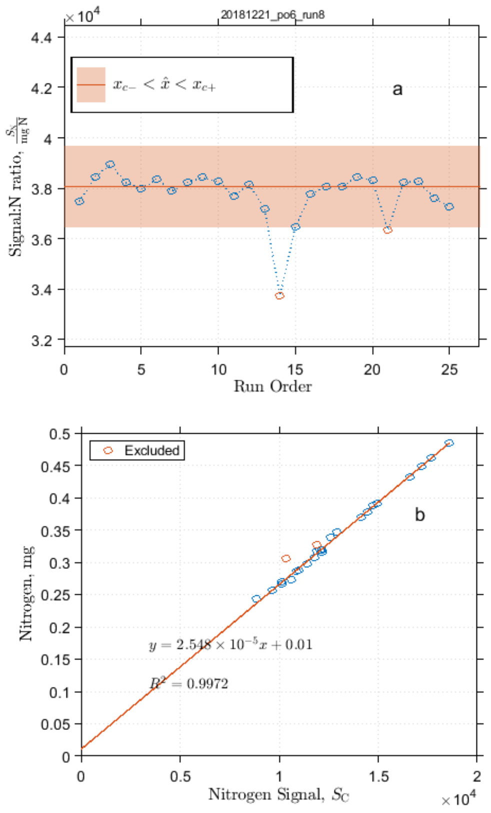


```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```

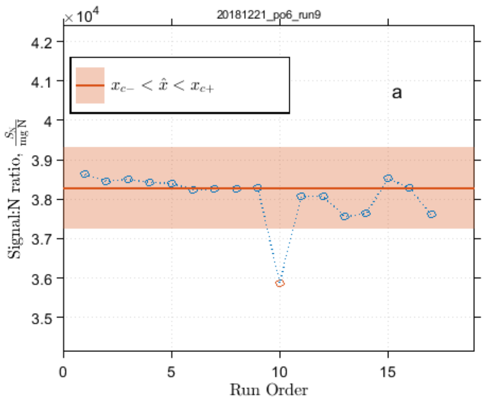


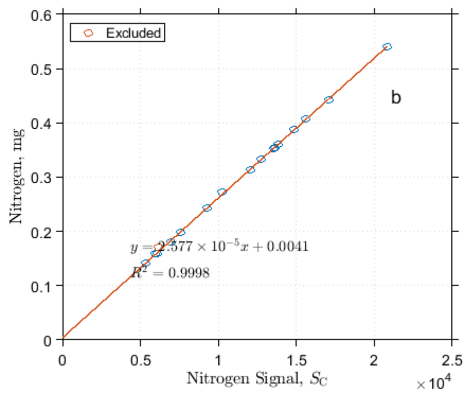


```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```

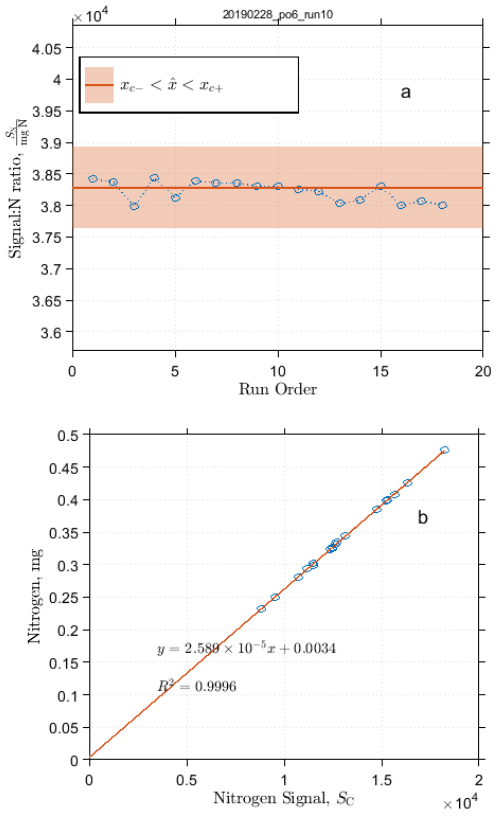


```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```

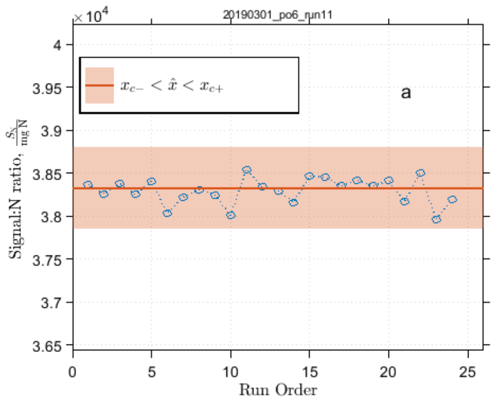


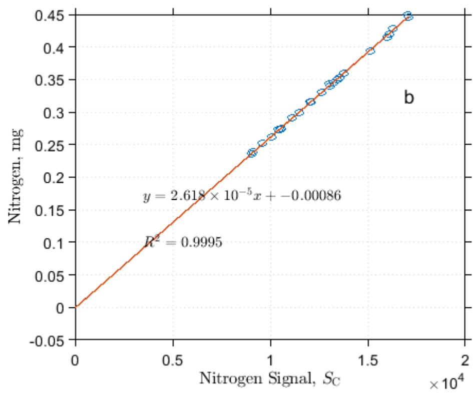


```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```

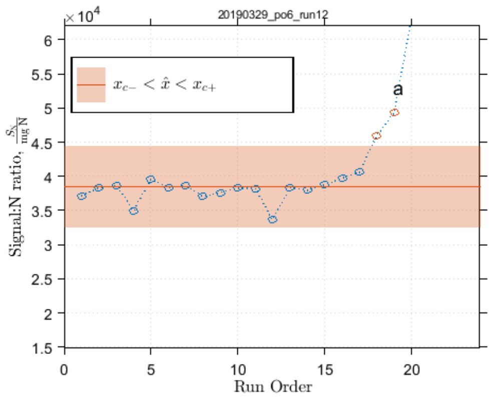


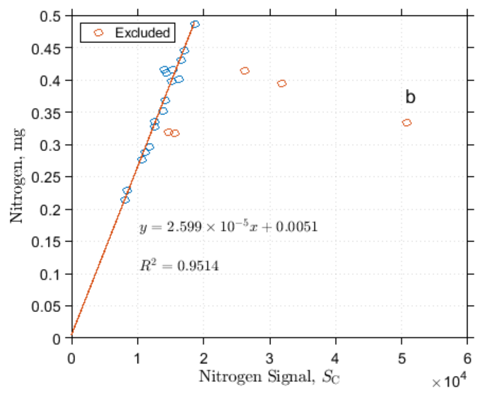
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



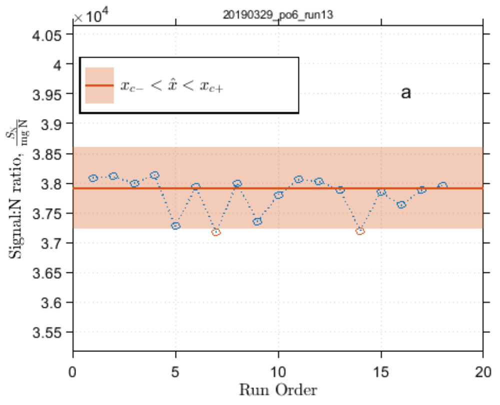


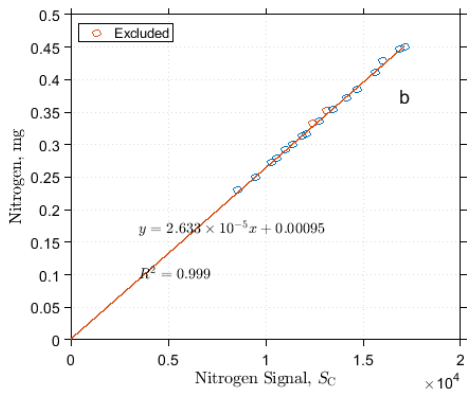
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```



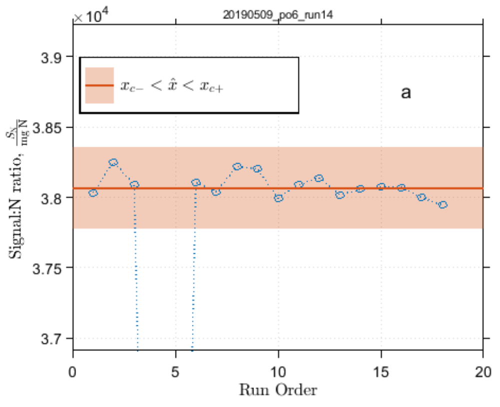


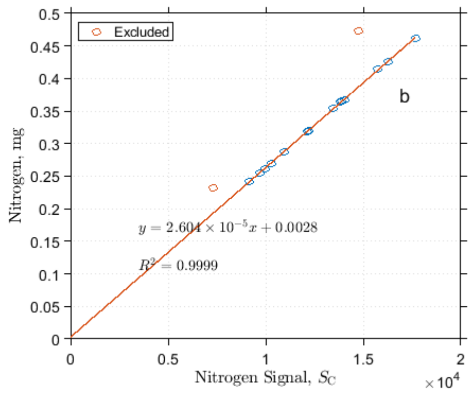
```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```





```
Warning: The DrawMode property will be removed in a future release. Use the
SortMethod property instead.
```





Carbon and Nitrogen in filter samples

```
            % Nitrogen

            if ~isnan(sampleNitroBlank)
                sampleNitroAreas = nitroAreas(sampleDex) - sampleNitroBlank; % Sample Carbon areas blank corrected (minus tin square foil)
            else
                sampleNitroAreas = nitroAreas(sampleDex) - mean(allNitroSampBlanks(allNitroSampBlanks > 0)); % Sample Carbon areas blank corrected (minus tin square foil)

            end

            %sampleCarbWeight = polyval(carbModel,sampleCarbAreas);  % calculate sample Carbon in mg

            % substracting the intercept of cal curve
            sampleNitroWeight = polyval(nitroModel,sampleNitroAreas) - nitroModel(2);  % calculate sample Nitrogen in mg

            % Nitrogen

    % tile figures

        if k == 1

            screenSize = get(0,'ScreenSize');

            plotWidth = 350;
            plotHeight = 325;

            nCols = floor(screenSize(3)/plotWidth);
            nRows = floor(screenSize(4)/plotHeight);

        end

        xPos = mod(k - 1,nCols) * plotWidth;
        if k > nCols
            yPos = (screenSize(4) - (67 + ceil(k/nCols) * plotHeight));
        else
            yPos = (screenSize(4) - (ceil(k/nCols) * plotHeight));
        end

%
```

```
        %% Export N concentration data to a file along with sample ID
        sampleID = str2double(runNames(sampleDex));

    % if sampleID are not being coverted to numbers try pass them directly

    if sum(isnan(sampleID)) == numel(sampleID)
        sampleID = cell2num(runNames(sampleDex));

    elseif any(isnan(sampleID))

        a = runNames(sampleDex);

        sampleID(isnan(sampleID)) = cell2num(a(isnan(sampleID)));
    end

if k == 22
    stop
end
        runNumber = repmat(k,numel(sampleID),1);

        M = [runNumber sampleID sampleNitroWeight];

    if k == 1
        dlmwrite('P06nitroWeigths.txt',M,...
        'delimiter',' ','roffset',0,'precision',9)

    else

        dlmwrite('P06nitroWeigths.txt',M,'-append',...
        'delimiter',' ','roffset',0,'precision',9)
    end
```

```
        %% Export calibration curve and other performance metrics to file

    runNumberStr = sprintf('runNumber_%d',k);

    runDataPN.(runNumberStr).nitroModel = nitroModel;
    runDataPN.(runNumberStr).calNitroBlankReps = calNitroBlankReps;
    runDataPN.(runNumberStr).sampleNitroBlankReps = sampleNitroBlankReps;

    if k == length(runFiles)
        save('P06PNrunDataStrct.mat','runDataPN')
    end
```

```
end
```

Export figures to file

```
  ag = findobj; % all graphical objects

  nf = ag(2).Number; %max(ag(ag==fix(ag))); % maximal integer value is number of figures


if strcmpi(exportFigs,'yes')

    cd([ homeStr '/Documents/CVO/Workshops/Writing/POCLatexVersion/Figures'])

       for i = 1:nf

           %print(i,'-dpdf',['Fig' num2str(i) 'PNCubeRun' rundID '.pdf'])
           print(i,'-dpng',['Fig' num2str(i) 'PNCubeRun' rundID '.png'])

       end

end
```

```
      disp('Process completed')
      toc
```

```
Process completed
Elapsed time is 226.891010 seconds.
```

Published with MATLAB® R2019b